

Adaptrade Builder

Version 5

User's Guide

Copyright © 2010 - 2026 Adaptrade Software
www.Adaptrade.com

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.

EasyLanguage and TradeStation are registered trademarks of TradeStation Technologies, Inc.

Last Revision: November 2026 (version 5.0.0)

Copyright © 2010 – 2026 Adaptrade Software
www.Adaptrade.com

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 any number of copies of the software on your devices, provided it is for your use only. A "single user" license permits the use of the software on no more than one device at a time. A "two-user" license permits the software to be run on two devices at the same time, and so on.
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 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

Software License Agreement ..... **iv**

Table of Contents ..... **v**

Introduction **1**

Overview *1*
Genetic Programming *3*
Build Algorithm *4*
Entry and Exit Conditions *5*
Order Types *8*
Neural Network Indicators *13*
Deep Neural Networks *15*
Trading Strategy Structure *18*
Example *20*

Getting Started **25**

Installation *25*
Program Layout and Configuration *27*
Working with Project Files *29*
Quick Start Steps *30*

Symbols and Settings **40**

Symbol Library *40*
Obtaining Price Data 41
Equity Curve Symbols 42
Adding a Symbol 44
Custom Indicators 47
Symbol Settings 49
Build and Evaluation Symbols *53*

Price Charts **59**

Creating and Using Price Charts *59*
Trading Orders *63*

Build Settings **65**

GP (Genetic Programming) Settings *65*
Build Metrics *68*
Stress Testing *72*
Lock Options *74*
Indicators and Order Types *75*
Strategy Logic Options *76*
Neural Network Indicator Settings *78*
Parameter Ranges *79*
Adaptrade Editor *80*
Deep Neural Network Settings *81*

Evaluation Options *85*
Position Sizing Settings *89*

Build Results **92**
Build Progress Window *92*
Messages Window *94*
Build Results Tables *95*
Performance Report Window *98*
Build Report Window *100*
Strategy Code Window *101*
Equity Curve Window *102*
Trade List Table *103*

Usage Topics **106**
Stress Testing and Monte Carlo Analysis *106*
Ideal Trades *110*
Out-of-Sample Performance *114*
Build Time *116*
Post-Build Testing and Optimization *117*
Common Questions *118*
Tips and Hints *123*

Ribbon Menu **127**
File Menu *127*
Build Menu *131*
Evaluation Menu *131*
Position Sizing Menu *131*
View Menu *131*
Clipboard Menu *134*
Ribbon Tab Buttons *135*

Appendix: Performance Metrics **136**

Appendix: Technical Indicators **146**

Appendix: Code Conventions **152**

Index **154**

Chapter 1

Introduction

Overview

Adaptrade Builder is a stand-alone Windows program that automatically generates trading strategies for several popular trading platforms, including TradeStation, MultiCharts, NinjaTrader 7/8, MetaTrader 4, and AmiBroker. Builder synthesizes unique combinations of trading indicators, trading rules, price patterns, entry and exit order types, and other user-selectable options and generates the corresponding strategy code. The program can generate an almost unlimited variety of trading strategies, including breakout, trend-following, stop-and-reverse, end-of-day, day trading, and others. All strategies created by Builder are designed around the user's specified performance requirements and goals and can be as simple or as complex as desired.

Builder can also be used to generate deep neural networks (DNNs) for price prediction using a process like that used to generate trading strategies. The program can generate DNNs with hundreds of thousands of nodes and train them using the same techniques used to train large language models (i.e., AI chatbots). The resulting neural networks can either be used as standalone prediction tools or incorporated into the strategy building process to become part of the strategy logic.

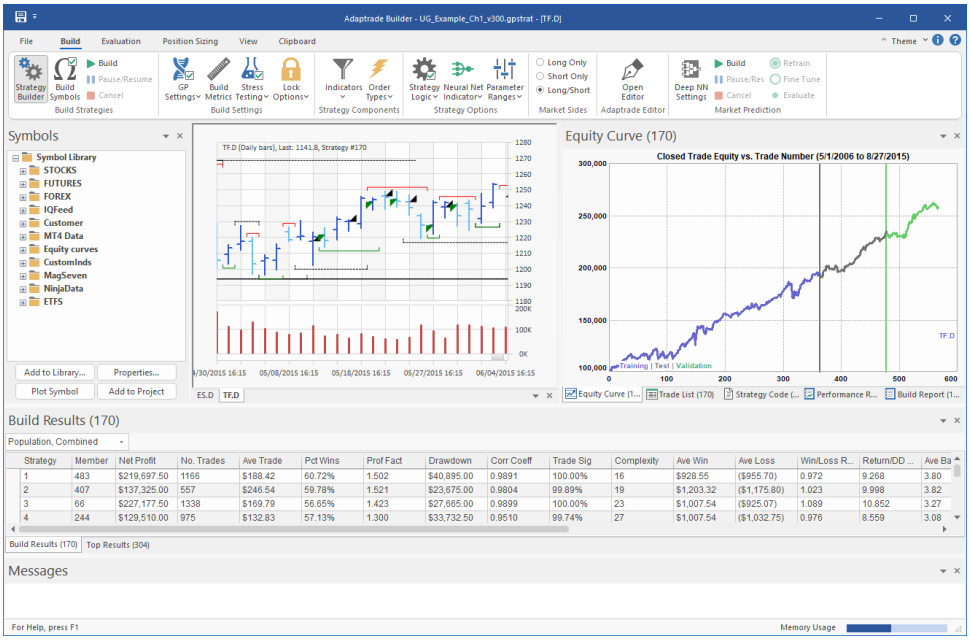


Figure 1.1. Main window of Adaptrade Builder.

Builder is designed to provide a superior alternative to 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 this traditional approach, a strategy is based on a market "hypothesis"-- 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 the traditional process automatically, as well as offering features and options not available in traditional strategy building, such as DNNs. The program generates an initial population of trading strategies by "seeding" the population based on the user's selected options. 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 and validation (“out-of-sample”) segments. Each candidate strategy is essentially a hypothesis that is either supported or refuted by the out-of-sample testing.

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 a suitable number of degrees-of-freedom to avoid both under-fitting and over-fitting. Out-of-sample testing and real-time tracking of the strategy results are the essential means used to verify these requirements.

Builder is designed to generate strategies for almost any market and time frame, from tick bars 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 copied to the trading platform for subsequent execution to generate trading signals. In addition, code is provided for TradeStation/Multicharts and NinjaTrader 7/8 to enable any DNN generated by Builder to be incorporated into any EasyLanguage or NinjaScript strategy, regardless of whether the strategy was generated by Builder.

Some of the user options in Builder include the code type (current versions of TradeStation/MultiCharts, TradeStation 2000i, NinjaTrader 7, NinjaTrader 8, MetaTrader 4, AmiBroker); specifying strategies as long-only, short-only, or combined long and short trading; requiring the long and short entry rules to be logical opposites; limiting entries and exits to specified times; limiting the number of entries per day for day trading strategies; including a neural network in the entry conditions; specifying one of several optional position sizing methods to include in the strategy code; building using stress testing/Monte Carlo analysis; including or 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.

Builder can also be used as a stand-alone trading platform for end-of-day trading. The program includes price charting, which displays the trade entry and exit points on the chart. A Trading Orders window displays the current open position(s) and any pending orders for the next bar. Used in conjunction with the data provider and brokerage of your choice, the generated strategies can be evaluated day-to-day in Builder and the orders sent manually to your broker for execution.

Genetic Programming

Builder uses a technique called genetic programming (GP),[1] which belongs to a class of machine learning algorithms 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, in which one member of the population is selected to be modified by randomly changing parts of its genome.

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, although any member of the final population may be selected.

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 100 to 500 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:

- The universe of possible trading rules and strategy logic is much larger than the universe of performance results. Therefore, it's easier to specify the desired results and let the computer find the rules and logic than to specify the rules and logic and hope to get the desired results.
- The GP process 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 out-performers 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 few days 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 as straight-forward as possible for any trader, individual or professional, who has a basic understanding of strategy trading and is familiar with one of the supported trading platforms.

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 markets 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.

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 selected markets, 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.

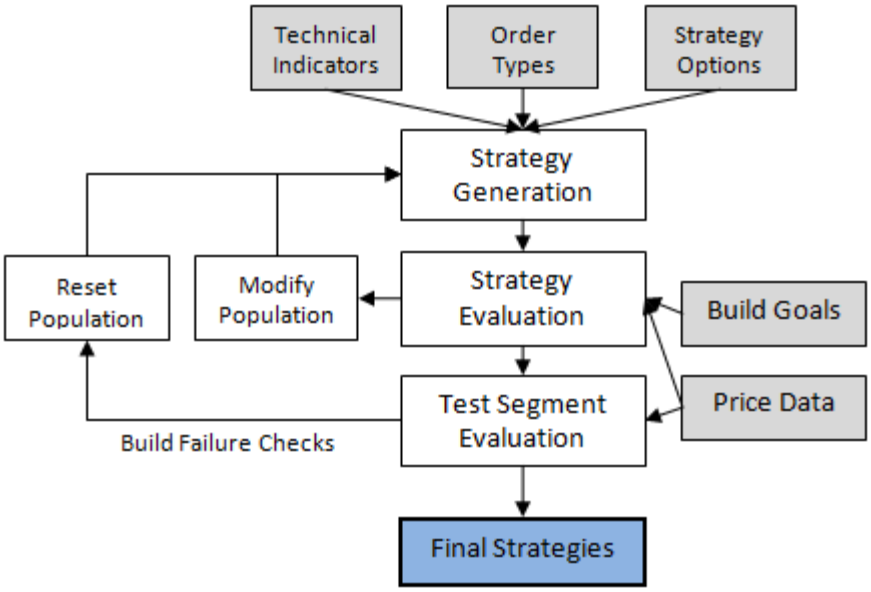


Figure 1.2. Build algorithm in Adaptrade Builder.

If the user has selected one of the build failure rules, the results on the test segment are checked after the 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 or the build termination conditions have been met, the strategies ("Final Strategies") are evaluated on the validation segment and the results are listed in the Build Results 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. Optionally, a neural network indicator can be included in the build process and evolved along with the other strategy features; see Neural

Network Indicators, below. Note that neural network indicators are different than deep neural networks (DNNs), which are discussed in a separate section.

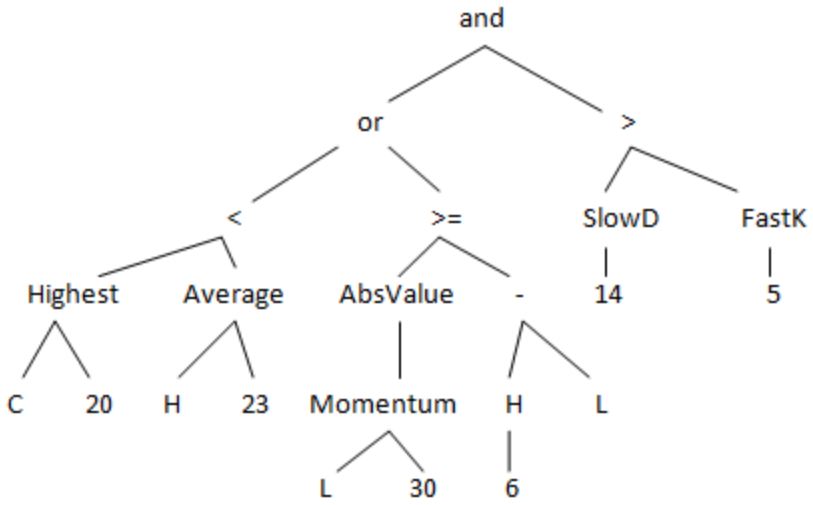


Figure 1.3. Entry condition example showing tree structure ((Highest(C, 20) < Average(H, 23) or AbsValue(Momentum(L, 30)) >= H[6] - L) and SlowD(14) > FastK(5)).

The entry and exit conditions are represented as tree structures, as shown above in Fig. 1.3. 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.

Note: As shown in Table 1 (see footnotes), not all indicators are available for all code types (scripting languages). For example, if the MetaTrader 4 code type is selected on the Evaluation Options tab prior to building, some indicators will be excluded from the build set and will not be included in the strategies. If a strategy previously built for TradeStation/MultiCharts includes one or more of these indicators and is then re-evaluated when the MetaTrader 4 code type has been selected, the resulting code will be invalid and will not compile in MetaTrader. Also, some indicators generate different values in different platforms, and Builder has been designed to duplicate these differences depending on the selected code type. For this reason, a strategy may produce different results if evaluated under a different code type than was selected for building the strategy.

The EasyLanguage code corresponding to the tree structure shown in Fig. 1.3 is listed below.

```
VarL1 = Highest(C, 20);
VarL2 =  Average(H, 23);
VarL3 = Momentum(L, 30);
VarL4 = AbsValue(VarL3);
VarL5 = H[6];
VarL6 = VarL5 – L;
VarL7 = SlowD(14);
VarL8 = FastK(5);
CondL1 = VarL1 < VarL2;
```

```
CondL2 = VarL4 >= VarL6;
CondL3 = CondL1 or CondL2;
CondL4 = VarL7 > VarL8
EntCondL = CondL3 and CondL4;
```

In this example, the long entry condition is given by the last statement, EntCondL.

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. 1.3, starting with ">" (i.e., SlowD(14) > FastK(5)), might be replaced with a different subtree from another member of the population. Each crossover operation produces two new trees as the subtrees from the parents are exchanged.

There are six different mutation operators: point mutation, subtree mutation, subtree simplification, subtree complication, subtree grow, and subtree prune. Point 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). Subtree mutation replaces a subtree with a new, randomly generated one. Subtree simplification replaces a subtree with a similar one that has fewer nodes. Subtree complication replaces a subtree with a similar one that has more nodes. Subtree grow adds a new subtree to the existing tree, and subtree prune removes a subtree from the tree.

Table 1. Indicators Available in Builder

Simple moving average	Directional movement index (DMI) [1, 2]
Exponential moving average	Average directional index(ADX)
Weighted moving average	True range (TR)
Triangular moving average[1, 2]	Average true range (ATR)
Adaptive variable moving average	Standard deviation
Zero-lag trend	Ulcer index
Adaptive zero-lag trend	Bollinger band
Moving average convergence divergence (MACD)	Keltner channel[1, 2]
Triple exponential moving average (TRIX) [1]	Lowest
Momentum	Highest
Rate of change (ROC) [1]	Volume
Fast K stochastic[1, 2]	Accumulation/distribution
Fast D stochastic	Chaiken oscillator[1]
Slow D stochastic	Crosses above/below[1]
Relative strength indicator (RSI)	Price patterns (O, H, L, C, O[N], etc.)
RocketRSI	Consecutive bars up/down
Inverse Fisher RSI	Congestion count
Inverse Fisher cycle	Typical price
Adaptive inverse Fisher RSI	Floor trader pivots
Adaptive inverse Fisher cycle	Day of week
Commodity channel index (CCI)	Time of day
Directional indicator (DI+/DI-)	Absolute value

[1]Indicator not available for MetaTrader 4 code. [2]Indicator not available for AmiBroker code.

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.

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 of price 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). This is called indicator nesting. Currently, nested indicators are not available for MetaTrader 4. Where available, nesting can be turned off in order to reduce strategy complexity using an option on the Strategy Logic window.

If multiple data series are included in the build, the semantic rules also ensure compatibility between the different series. For example, if the primary series (the series on which trades are placed) is AAPL and a secondary series is SPY, indicators can be applied to either symbol, but comparisons between the symbols are prevented. However, comparisons are allowed between data series if they have the same symbol but different timeframes, such as between 30-minute bars and daily bars of the E-mini S&P futures (ES).

Order Types

Entry Orders

The following types of entry orders are available in Builder:

- Enter at Market
- Enter on Stop ($)
- Enter on Stop (%)
- Enter on Stop (Price)
- Enter at Limit ($)
- Enter at Limit (%)
- Enter at Limit (Price)

Entry orders are placed when the entry conditions, as described in the previous section, are true. 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.

The dollar-based stop and limit entries ($) use a fixed-size deviation from the current bar's close based on a currency value (e.g., dollars for accounts denominated in dollars). The size of the stop or limit value chosen by the program during building is based on the "Fixed Entry Size, Stops/Limits" parameter range entered on the Parameter Ranges window. For example, if the dollar range is from 100 to 750, the program may select any value within that range

during the build process. The selected value is divided by the point value for the symbol to obtain the number of points. For example, a stop value of $500 means the entry price is calculated to be $500 from the current bar's close per share/contract. For the E-mini S&P futures, for example, this means the entry stop would be 10 points above the close for a long trade because each point is worth $50 for the E-mini.

Percentage-based stop and limit entries (%) are set at a percentage of the current bar's close above (below) the current bar's close for a long (short) trade. For example, if the close is 25, a 5% limit entry would be placed 1.25 points below the close for a long trade. The percentage value is chosen from the "Percentage Entry Size, Stops/Limits" parameter range on the Parameter Ranges window.

Price-based 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), WAverage(price, N), TriAverage(price, N), ZLTrend(price, N), DayPrice, BollingerBand, KeltnerChannel, TypicalPrice, AS_PivotS, AS_PivotR, or any available custom indicator returning a price value;
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);
AS_PivotS is the floor trader pivot support price, which can be at one of three levels, depending on the function input value;
AS_PivotR is the floor trader pivot resistance price, which can be at one of three levels, depending on the function input value;
`Fr` is a constant; and
`PriceDiff` is one of: true range, ATR, $\text{PriceValue}_1 - \text{PriceValue}_2$, or $\text{AbsValue}(\text{PriceValue}_1 - \text{PriceValue}_2)$, in which PriceValue_1 and PriceValue_2 are instances of PriceValue, as described above.

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 N's are chosen from the "Lookback" ranges (Indicators, Price Patterns) entered on the Parameter Ranges window.

Fr is either a constant that is multiplied by PriceDiff or a fixed number of points. In the latter case, there is no PriceDiff, and Fr is simply added to or subtracted from PriceValue. When used as a multiplier, it is selected from the "Price Difference Multiple, Entries" range entered on the Parameter Ranges window. When used as a fixed number of points, the number of points is derived from the "Fixed Entry Size, Stops/Limits" range on the Parameter Ranges window.

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))

EntryPrice = Highest(C, 15) + 35
```

```
EntryPrice = BollingerBand(C, 25, 1.2)
```

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[3] - 4.3*AbsValue(XAverage(L,5)- Xaverage(C,2))

EntryPrice = Average(L, 5) - 74

EntryPrice = MyCustomIndicator(L, 35)
```

In the last example, MyCustomIndicator represents a custom indicator function provided in the price data file; see Symbols and Settings, Symbol Library, Custom Indicators. These examples 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 are examples of each type of entry order:

- Enter at Market: buy next bar at market (i.e., buy next bar at the open of the next bar).
- Enter on Stop ($): buy next bar at close + 5 stop (i.e., buy 5 points above this bar's close).
- Enter on Stop (%): sell short next bar at close * (1 - 0.02) stop (i.e., sell short 2% below this bar's close).
- Enter on Stop (Price): buy next bar at Average(C, 18) + 1.3 * TrueRange stop (i.e., buy at the 18-bar average close plus 1.3 times the bar's true range).
- Enter at Limit ($): buy next bar at close - 10 limit (i.e., buy 10 points below this bar's close).
- Enter at Limit (%): sell short next bar at close * (1 + 0.035) limit (i.e., sell short 3.5% above this bar's close).
- Enter at Limit (Price): buy next bar at L - Average(H - L, 10) limit (i.e., buy at the low minus the 10-bar average high-low range).

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

- Exit at Target ($)
- Exit at Target (%)
- Exit at Target (Price)
- Trailing Stop ($ Floor)
- Trailing Stop (ATR Floor)
- Protective Stop ($)
- Protective Stop (%)
- Protective Stop (Price)
- Exit After N Bars
- Exit After N Bars Profit
- Exit After N Bars Loss
- Exit After Time
- Exit at Market
- Exit End-of-Day
- Exit End-of-Week

Target Exits

The three target exits use limit orders to exit the trade at a price that represents a profit for the trade. The first target exit type ($) uses a fixed-size target based on the value of the profit target (e.g., dollars for accounts denominated in dollars) and is applied per share or contract. For example, a target size of $500 means the target price is calculated so that if the target is hit, the profit will be $500 per share or contract before trading costs. For the E-mini S&P futures, for example, this means the stop size would be 10 points above the entry price for a long trade because each point is worth $50 for the E-mini. For a stock trade, regardless of the number of shares, a $2 target would be placed two points above the entry price for a long trade or 2 two points below the entry for a short trade. The size of the target chosen by the program during building is based on the "Fixed Target Size" parameter range entered on the Parameter Ranges window.

Percentage targets are set at a percentage of the entry price above (below) the entry price for a long (short) trade. For example, if the entry price is 25, a 5% target would be placed 1.25 points above the entry for a long trade. The percentage value is chosen from the "Percentage Target Size" parameter range on the Parameter Ranges window.

Price-based target exits are constructed the same way as limit entry orders. Specifically, the price for this type of target exit is calculated as follows:

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

where `PriceValue`, `Fr`, and `PriceDiff` are as defined above for entry orders. One difference is that `PriceValue` can be optionally set to `EntryPrice`, which is the entry price for the trade. When used as a multiplier in exit orders, `Fr` is selected by the program from the "Price Difference Multiple, Exits" range on Parameter Ranges. When it represents a fixed number of points, `Fr` is derived by the program from either the "Fixed Stop Size, Exits" range or the "Fixed Target Size" range on Parameter Ranges, depending on whether the order type is a stop or limit exit.

For example, the following could be the price for a short target exit:

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

ExitPrice = BollingerBand(C, 17, -2.5)

ExitPrice = EntryPrice - 12
```

These could also be the exit prices for long protective stop exits.

Trailing Stops

Trailing stops in Builder are activated when the open profit on a closed-bar basis is above a threshold called the *floor*. There are two types of trailing stops in Builder: ones with fixed-size floors ($ Floor), and ones where the floor is calculated as a multiple of the average true range (ATR Floor). For fixed-size floors, the size of the floor is selected by the program from the "Fixed Stop Size, Exits" parameter range entered on the Parameter Ranges window. For ATR-based floors, the multiple of the ATR is selected from the "Price Difference Multiple, Exits" range on Parameter Ranges. 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 percentage of profit to lock in is chosen by Builder from the range 0 to 100 percent.

Protective Stops
As with target exits, there are three types of protective (also known as money management) stops. The fixed-size protective ($) stops use a fixed value (e.g., in dollars for accounts denominated in dollars), which is applied per share or contract. For example, a stop size of $500 means the stop price is calculated so that if the stop is hit, the loss will be $500 per share or contract. For the E-mini S&P futures, for example, this means the stop size would be 10 points below the entry price for a long trade because each point is worth $50 for the E-mini. For a stock trade, regardless of the number of shares, a $2 stop would be placed two points below the entry price for a long trade. The size of the stop is based on the range entered on the Parameter Ranges window ("Fixed Stop Size, Exits").

Percentage (%) protective stops are set at a percentage of the entry price below the entry price. These types of stops are often used for stock trading. For example, if the entry price is 25, a 5% protective stop would be placed 1.25 points below the entry for a long trade. The percentage value is chosen from the "Percentage Stop Size, Exits" setting on the Parameter Ranges window.

Price-based protective stops are constructed the same way as price-based target exits. In fact, the only difference is that the sign is reversed preceding the PriceDiff portion of the ExitPrice equation presented above.

Exiting After N Bars
The “Exit After N Bars” order type 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 during the build process. The "Exit After N Bars Profit" and "Exit After N Bars Loss" order types work similarly. With the "Exit After N Bars Profit" order type, the trade is exited at the open of the next bar if the number of bars since entry is greater than or equal to N *and* the close of the bar is greater (less) than the entry price for a long (short) trade; i.e., if the trade is profitable before costs.

Likewise, with the "Exit After N Bars Loss" order type, the trade is exited at the open of the next bar if the number of bars since entry is greater than or equal to N *and* the close of the bar is less (greater) than the entry price for a long (short) trade; i.e., if the trade is unprofitable before costs. Normally, N is selected from the "Exit-After Number of Bars" setting on the Parameter Ranges window. However, for intraday strategies for which the end-of-day exit has been selected, N is limited to the number of bars in the trading session.

Other Types of Market Exits
The “Exit After Time” order type 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”. Note that this implies the time stamp for the trade exit will be the time of the bar following the bar where the exit is triggered. In the preceding example, if the bars are 30 minute bars, the exit time would be shown as 11:00.

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

The “Exit End-of-Day” order type 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. *For TradeStation and MultiCharts, this exit type is primarily for back-testing purposes. It can be problematic because the market will be closed by the time the order is sent, which means it may be placed in the after-hours market as a limit order and therefore may not be filled. To achieve end-of-day exits in real-time*

trading on intraday data in any of the supported languages, the optional time-based exit on the Strategy Logic window should be used. The End-of-Day exit type is not available for MetaTrader 4 or AmiBroker. For NinjaTrader, this exit only applies to intraday data.

The “Exit End-of-Week” order type is essentially the End-of-Day exit applied on a Friday. It causes the trade to exit at the close on Friday. This order type is only available for EasyLanguage code (TradeStation/MultiCharts), which is the only supported language that includes an end-of-session order type ("SetExitOnClose"), other than NinjaTrader, which does not allow the end-of-session exit to be restricted to a specific day. Please note that the exit occurs only on a Friday, so if there are no bars on Friday, there will be no exit that week via this order type. *As with the End-of-Day exit, this exit type can be problematic because the market will be closed by the time the order is sent, which means it may be placed in the after-hours market as a limit order and therefore may not be filled. As an alternative to achieve end-of-week exits in real-time trading on intraday data in any of the supported languages, the optional time-based exit on the Strategy Logic window can be used with the optional day-of-week restriction.*

Neural Network Indicators

Adaptrade Builder includes two different types of neural networks. This section describes a simple type of neural network with parameters (weights) that are defined as inputs to the trading strategy and whose values are determined during the strategy building process. Since there are practical limits to the number of strategy inputs, these neural networks are small and tend to function as composite indicators. In effect, they combine traditional indicators, such as moving averages, momentum, and stochastics, into a single value that is then used to filter the trading signals from the other entry conditions. The other type of neural network available in Builder is a so-called deep neural network (DNN), which is more complex, can contain hundreds of thousands of parameters, is trained separately, and is discussed in the section Deep Neural Networks.

In general, neural networks are nonlinear models loosely based on the way the brain is structured as a network of interconnected neurons. A neural network is established by fitting it to a given set of input and output data with the goal of accurately predicting new outputs from input data not used in the fitting process. A successful neural network is one that generalizes well to new input data.

The neural network indicators available in Builder are so-called feedforward neural networks, which are often described graphically as a set of nodes connecting an input layer through a "hidden" layer of nodes to one or more outputs. The inputs are summed together using a set of weights that connect the nodes to produce the output. The weights are determined by the fitting or *training* process. A feedforward neural network with a single output and one hidden layer is shown below in Fig. 1.4. Neural network indicators in Builder are limited to one hidden layer.

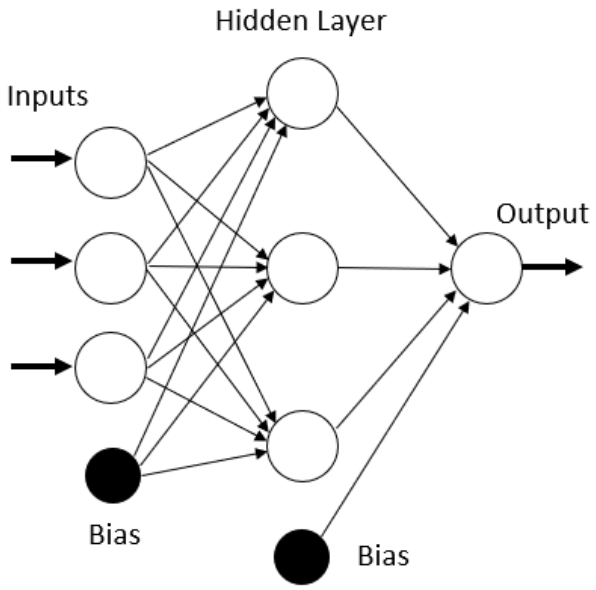


Figure 1.4. A feedforward neural network with a single output and one hidden layer. It includes three input nodes, three nodes in the hidden layer, and two bias nodes.

The output from the network is defined by the following equations:

$$H_1 = \text{ReLU}(w_{11} * I_1 + w_{21} * I_2 + \ldots + w_{n1} * I_n + B_1)$$

$$H_2 = \text{ReLU}\ (w_{12} * I_1 + w_{22} * I_2 + \ldots + w_{n2} * I_n + B_2)$$

…

$$H_m = \text{ReLU}\ (w_{1m} * I_1 + w_{2m} * I_2 + \ldots + w_{nm} * I_n + B_m)$$

$$\text{Output} = \tanh(w_{nm+1} * H_1 + w_{nm+2} * H_2 + \ldots + w_{nm+m} * H_m + B_o)$$

The I_i represent the inputs. In the context of trading, these could be anything that might have some predictive value for trading, such as momentum, stochastics, ADX, moving averages, etc. The H_j represent the hidden nodes, the weights are given by the w_{kl}, B_j are the weights of the bias nodes for the hidden layer, B_o is the weight of the bias node for the output layer, and the output value is given by Output. Bias nodes always have a value of 1 and so are represented by their respective weight values only. ReLU is the so-called rectified linear unit, which is defined by the function f(x) = max(0, x). The hyperbolic tangent function, tanh, returns a value in the range -1 to +1, so the output will lie in this range. The inputs are scaled so that they also lie between -1 and +1.

Training the network to determine the weight values involves iterating on the weights using known input-output data. The technique traditionally used to adjust the weights in an iterative manner is called back-propagation, although any similar approach may be used, such as a genetic algorithm.

In Builder, the inputs to the network are chosen by the genetic programming process and evolved along with the entry and exit conditions using the same processes of crossover and mutation used to evolve the entry and exit conditions. The output of the network allows for a long entry if it's greater than or equal to 0.5 and for a short entry if it's less than or equal to -

0.5. This condition is in addition to any existing entry conditions (see Trading Strategy Structure, below). For example, if there is a long entry condition, it must be true *and* the neural network output must be at least 0.5 for a long entry.

As mentioned above, the inputs to the neural network are scaled so that they lie between -1 and +1. The scaling is performed on a moving or trailing basis over the most recent NNLookBack bars. In other words, the minimum and maximum values of each input over the most recent NNLookBack bars are used to scale each input so that it lies between -1 and +1. The value of NNLookBack is a user-selectable option with a default value of 100.

In addition to evolving the inputs to the network, Builder also evolves the number of nodes in the hidden layer and the weight values. The initial number of inputs is a user-selectable option, along with the maximum number of nodes in the hidden layer. The number of inputs may change from strategy to strategy over successive generations as the inputs from different strategies are combined using crossover.

The total number of weights in the network will be given by (n + 2) * m + 1, where n is the number of inputs and m is the number of nodes in the hidden layer, provided m is at least one. If there is no hidden layer (i.e., m is zero), the number of weights is the same as the number of inputs plus 1 (for the bias). In the resulting strategy code, each weight is listed as a strategy input. Other strategy inputs are added as necessary for the neural network inputs, such as the look-back length of a moving average.

To include a neural network indicator, simply check the option on the Neural Net Indicator window and select the available settings, such as the initial number of inputs and the maximum number of hidden nodes. To develop strategies in which the neural network indicator provides the only entry condition, set the tree depth (GP Settings) to zero, which will ensure that the usual entry conditions are always "true".

Deep Neural Networks

In addition to the neural network indicators described in the previous section, Builder also enables you to build populations of so-called deep neural networks (DNNs). The word "deep" refers to the fact that these networks typically have more than one hidden layer, and they often contain hundreds of thousands of parameters (weights). Unlike the neural network indicators, these DNNs are trained separately in their own populations. Once built, a DNN can be saved and selected to be included in strategies as part of the regular strategy building process.

Generally, DNNs are powerful general-purpose function approximators. In Builder, DNNs are trained to predict the closing price N bars ahead, where N is a user-selected parameter. A key feature of DNNs is that unlike traditional systematic trading methods, a deep neural network is less likely to overfit the training data, even when there are many more weights than data points.

The DNNs in Builder are long short-term memory (LSTM) networks. An LSTM is a specific type of a so-called recurrent neural network, which is intended for processing sequential data, such as price bars, and contains connections that feed back to the same node or to previous nodes. An LSTM is a more advanced version of a standard recurrent network, designed to avoid some of the limitations of the basic version. For additional details, including references to articles discussing the use of LSTM networks in strategy trading, please refer to the article Deep Neural Networks for Price Prediction, Part 1 on the adaptrade.com website.

The genetic programming process of Builder is used to evolve the inputs to the DNN, where the inputs are chosen from the set of available indicators; see Indicators and Order Types in the Build Settings chapter. By constructing a population of DNNs, each with a different set of randomly chosen inputs, the GP algorithm can be used to find the best combination of inputs by evolving the population over successive generations.

The process of evolving a population of DNNs in Builder involves two levels of optimization. The GP process optimizes the inputs to the LSTM while the training process for each LSTM optimizes the weight values of each network. Once a new set of inputs is chosen based on the GP operators of mutation and/or crossover, the resulting LSTM is trained using either stochastic gradient descent (SGD) -- specifically, Adam -- or a neuroevolutionary algorithm (i.e., a genetic optimizer). Adam is a variation of SGD that uses an adaptive learning rate. The choice of training method is up to the user. In general, Adam converges faster and uses less memory but is more volatile. The genetic optimizer is guaranteed by design to converge steadily, if slowly, and may have a better chance of finding a global optimum.

When building trading strategies in Builder, the user can select from a set of more than 100 different performance metrics to define the fitness, which guides the evolution of the population. However, for building a population of DNNs, the existing metrics, such as net profit, drawdown, and so on, wouldn't be applicable since the DNN is not a trading strategy. Accordingly, a separate set of metrics was developed specifically for the evaluation of neural networks that predict prices.

The most basic of these is the mean squared error (MSE), which is the sum of the squared differences between the actual price and the predicted price divided by the number of predictions. This is the error measurement most used to train a neural network with a linear output, such as the LSTM networks implemented in Builder. Another key metric is Net Points, which is the total number of price points that could be accumulated if you bought on the open of the next bar and sold on the close of the bar N bars in the future for an N-bar price prediction if the prediction is for a positive price change (or sold short and bought back for a negative price change prediction). It's meant to provide an estimate of the profit potential of following the DNN's prediction. The Dir Correct metric calculates the percentage of predictions that are directionally correct; e.g., if the price change is positive and the prediction was for a positive price change, the direction is correct. All the metrics available for building and training DNNs are listed and described in the appendix Performance Metrics; the metrics for DNNs are listed at the end of the appendix. As with building trading strategies, the user can combine the different metrics to create a custom fitness measure that helps guide the build process for DNNs.

The basic process for building a population of DNNs is like that for building a population of trading strategies:

1. Select the symbol or symbols you want to build the DNN for on the Build Symbols window.
2. Select the genetic programming (GP) settings on the GP Settings window, including population size and number of generations.
3. Remove any indicators from the Indicators list that you don't want in the DNN inputs or select any to include in the DNN inputs.
4. Most of the options on the Strategy Logic window don't apply; however, you can deselect the option to use nested indicators if desired.
5. Verify that the parameter ranges are correct for your symbol(s) under the Parameter Ranges window.

6. Open the DNN Settings window and enter the settings that define the DNNs you want to build, such as the number of inputs, number of hidden layers, number of nodes per layer, prediction length (in bars), and so on. These options are explained in detail in the Build Settings chapter in Deep Neural Network Settings.
7. Finally, close the DNN Settings window and click the Build button on the Market Prediction panel of the Build menu.

There are a few things to keep in mind when building DNNs. First, most of the options available for building trading strategies also apply to DNNs. For example, you can build over multiple markets, where each symbol is identified as "primary". This means the DNN will be applied to each symbol and the training error calculated over all symbols, analogous to running a trading strategy over multiple symbols and recording the performance over the portfolio. Similarly, you can select some symbols to be secondary symbols, such as Data2, Data3, etc. In this case, the inputs may include references to the primary symbols as well as to the secondary symbols, such as "Average(C, N) of Data2".

Stress testing can also be used when building DNNs, although keep in mind that doing so will substantially increase the build times. Likewise, it's generally recommended to use small populations when building DNNs -- as small as 10. Remember that the GP process is only used to evolve the DNN inputs, not to train the networks, so a large population may not provide much additional benefit compared to a small population, and the build times will be substantially longer.

Build options related to trading strategies generally don't apply when building DNNs. For example, the Order Types list does not apply. Similarly, the Build Metrics window is not applicable since those metrics don't apply to DNNs, and the Neural Net Indicator window will also have no effect on the DNN build process -- neural network indicators are a separate option, and you can't have a neural net indicator and a DNN in the same strategy. However, the code type and position sizing options both apply to DNNs for the reasons explained next.

After the DNNs are built, Builder applies each one to a simple strategy in which a long trade is opened if the DNN predicts the market will be higher in N bars, where N is the prediction length selected on the DNN Settings window, and exits the trade at market after N bars. Likewise, if the DNN predicts the market will be lower, it opens a short trade and covers it after N bars. This is intended to provide a rough measure of the DNN's performance as a supplement to the DNN-specific metrics discussed above. The code for each of these simple strategies is displayed in the code pane, and the performance of each one is shown in the performance results.

Keep in mind that these simple strategies are not intended to be traded; they only serve as an approximate measure of the DNN's performance. Likewise, since the DNNs will not typically be utilized in a strategy in the way that these simple strategies use them, it may be that a DNN with relatively poor performance in the simple strategy can still be effective when included in the regular strategy building process, provided the DNN metrics are positive.

After building a population of DNNs, they can be included in trading strategies. To include a DNN in a strategy in Builder, first save the DNN you want to include. To do this, right-click on the DNN in the Build Results table and select the option "Save Deep Neural Networks". This will save the DNN in a special file with the extension ".gpdnn". The corresponding file of weight values from the training process will be saved in a separate file, the name and location of which are stored in the gpdnn file. Next, reopen the DNN Settings window and select the option "Include DNN in strategies". Use the Select DNN button to select the DNN file you just saved or any other previously saved DNN file. Finally, Build a population of

trading strategies as you normally would.

If you later want to exclude the DNN from a strategy that was built with it, you can uncheck the option to include the DNN in strategies and re-evaluate the strategy from the Evaluation menu.

Although it's not specifically part of Builder, you can also include a DNN generated by Builder in a strategy you may have written yourself outside of Builder. To do this, follow the code pattern from a strategy generated by Builder that includes a DNN. For example, EasyLanguage code for Builder-generated strategies that include a DNN includes two "external" statements for the dll functions that the strategy uses to initialize and call the DNN. These functions -- DNNInit and DNNInference -- must be available to your strategy via the dll in order to execute the function calls. In addition, your strategy must define and evaluate the DNN inputs. You can copy the code for these parts of your strategy from the corresponding code from the simple strategy generated at the time the DNN was built, provided you saved the project file.

Please refer to the section Deep Neural Network Settings in the Build Settings chapter for details on the available build settings for DNNs. This section also explains the Retrain, Fine Tune, and Evaluate options on the Market Prediction panel of the Build menu.

Please note that Builder-generated strategies that include a DNN require a dll library that includes the functions called by the strategy code to initialize and evaluate the DNN. This dll library is normally installed as part of the program installation process and must be available in the location specified in the strategy code. Also, if the file of weight values is moved or is no longer available to the strategy (e.g., if the strategy is run on a different computer than the one on which the DNN was built), the path to the weight file in the DNNInit dll function call in the strategy code must be updated and/or the weight file copied to a new location so that the DNNInit function will be able to access the DNN weights.

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 = …

[Neural network inputs]
[Neural network output function]

If [position is flat and] LongEntryCondition is true
   [and neural network indicator >= 0.5]
   [and DNN output - close > threshold] then
      Long entry order…
      Initialize long exit orders as necessary…
```

```
If [position is flat and] ShortEntryCondition is true
   [and neural network indicator <= -0.5]
   [and close - DNN output > threshold] 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. Either the neural network indicator or the DNN can be included but not both.

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. 1.3. A long entry order is placed if the long entry condition is true, subject to the optional conditions, such as that the position is currently flat (out of the market) and that the neural network indicator or DNN value is above the threshold value. Likewise, a short entry order is placed if the short entry condition is true, subject to the optional conditions. 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 Order Types window in order to be included in the strategy.

The meaning of common input variables and order labels used in the strategy code is explained in Appendix: Code Conventions.

Example

As an introductory example, Builder was run on daily bars of a stock index futures market. The performance metrics chosen to guide the process are shown below in Fig. 1.5. In addition to maximizing the metrics shown under Build Objectives, specific requirements ("Build Conditions") were set for the correlation coefficient, trade significance, and the number of trades. The build metrics define the fitness function that's used to drive the evolution of the population.

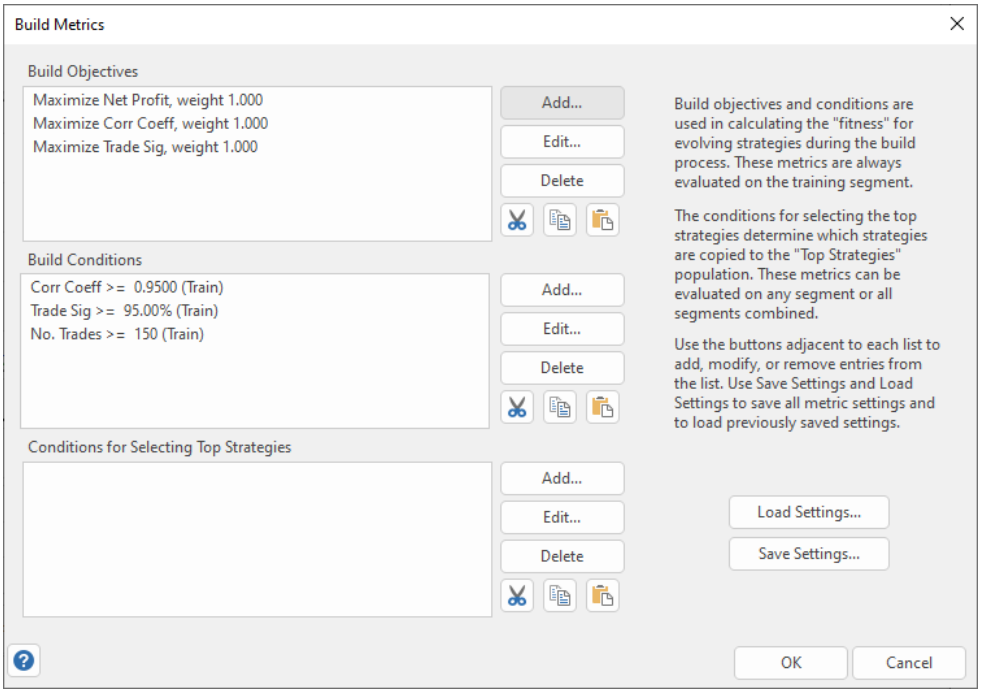


Figure 1.5. Build metrics for example build.

The genetic programming (GP) settings are shown in Fig. 1.6. The population size was set to 500 with the fittest 100 strategies saved after building. At the end of the 20th generation, the Build Failure Rules were evaluated. These rules help determine if the build process is proceeding normally or if the results are being over-fit. If either of the checked rules applies, the build process will be restarted. After 30 generations, the Build Termination rules were evaluated after each generation. These rules help determine if the build process is still improving the results or if the population has converged. If either of the checked rules applies, the build process will be stopped.

The train/test/validation division of data was set to 65% training, 20% test, and 15% validation. The build process ran over a total of 32 generations before one of the build termination rules stopped the build. To illustrate how the results evolved during the build, the Build Termination Tracking plot from the Build Progress window is shown in Fig. 1.7. The curve labeled "Train Fit" shows how the fitness on the training segment changed with each generation. The curve labeled "Test Fit" illustrates the fitness on the test segment, including a moving average defined by the settings made under Build Termination Options (Fig. 1.6).

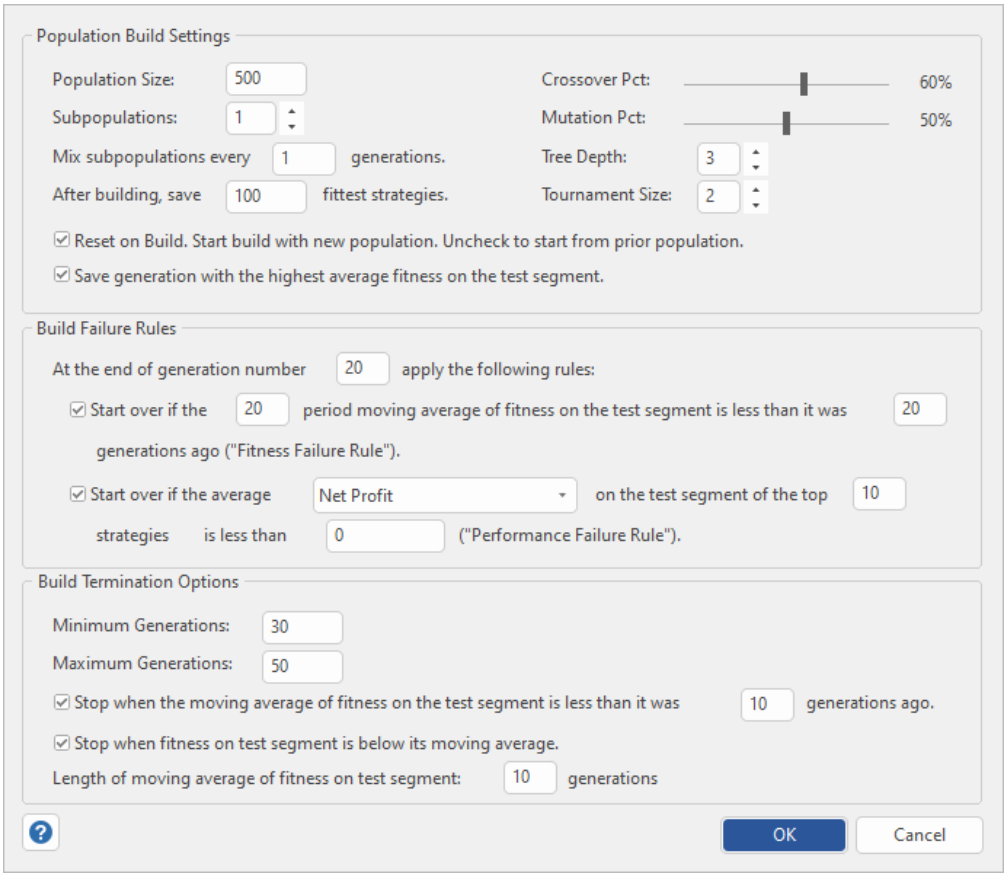


Figure 1.6. Genetic programming settings for example build.

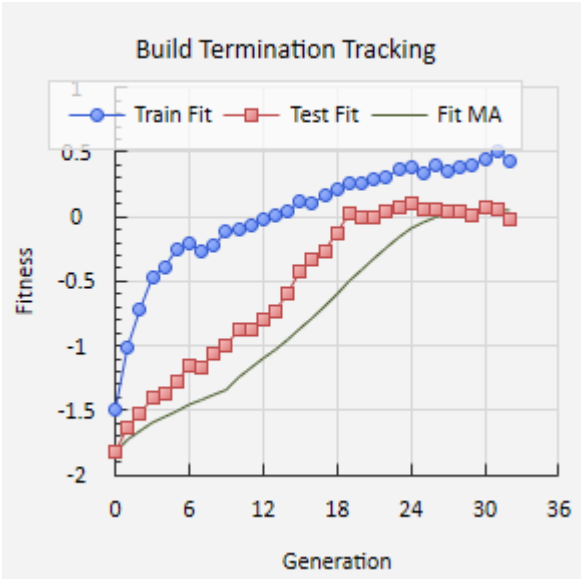


Figure 1.7. Fitness on the training and test segments during the build process.

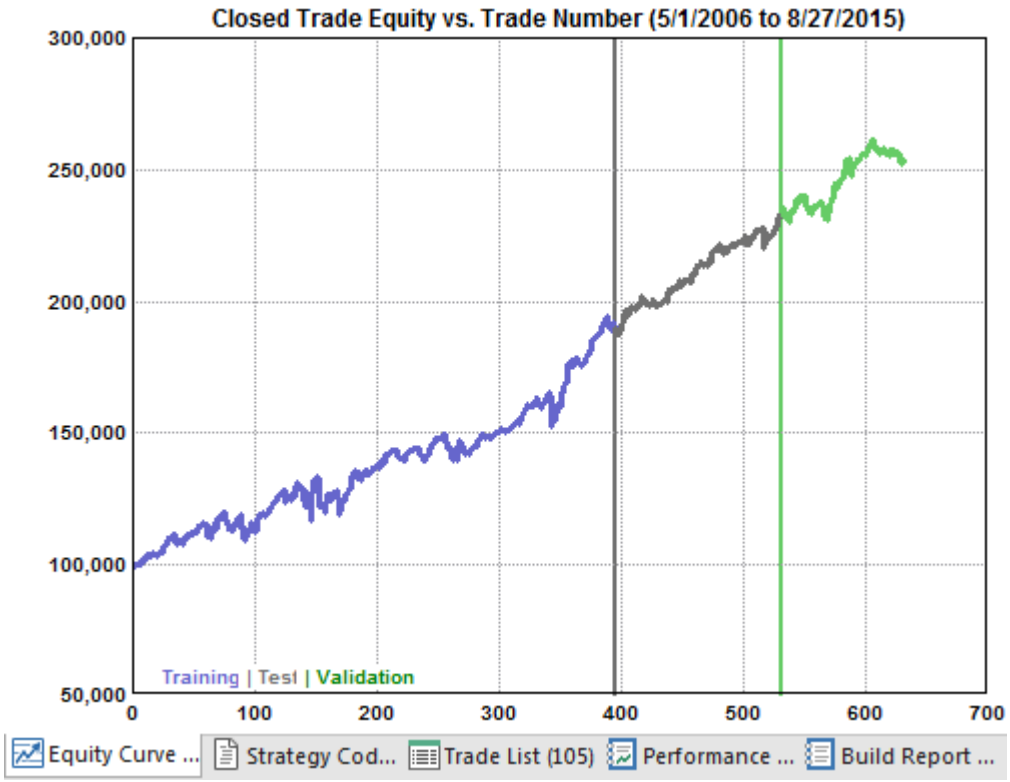


Figure 1.8. Closed trade equity curve after build.

As shown in Fig. 1.7, the fitness on the training segment generally increased throughout the build process, as expected, given that the GP process is designed to improve the results on the training data. The fact that the fitness also increased on the test segment, which is not used to evolve the population, indicates a good quality build. When the fitness on the test segment started to drop off, indicating that the strategies might be starting to over-fit the training data, the build process was stopped by one of the build termination rules. The generation with the highest fitness on the test segment was saved for the final results according to the option shown in Fig. 1.6.

After the build process was completed, the population results were sorted by the fitness on the test segment. The equity curve for one of the strategies with the highest fitness on the test segment is shown above in Fig. 1.8, indicating positive results on the validation segment. Finally, the EasyLangauge (TradeStation) code for the corresponding strategy is listed below.

```
{------------------------------------------------------------------------------
 Trading Strategy Code
 Population member: 105
 Max bars back: 87

 Created by: Adaptrade Builder version 2.1.0.0
 Created:    11/4/2016 5:31:20 PM

 Scripting language: TradeStation 6 or newer
 Price file:
 Build dates: 5/1/2006 to 5/22/2012
```

```
 Project file:  UG_Example_Ch1_v210.gpstrat
-------------------------------------------------------------------------------}
{ Strategy inputs }
Inputs: NL1 (4),         { Day of week (0 to 6 for Sunday to Saturday), long trades }
        NL2 (36),          { Indicator look-back length (bars), long trades }
        NL3 (17),          { Indicator look-back length (bars), long trades }
        ShiftL1 (7),       { Indicator shift value (bars), long trades }
        NS1 (7),           { Indicator look-back length (bars), short trades }
        NS2 (50),          { Max look-back length for adaptive indicator (bars),
short trades }
        NS3 (43),          { Indicator look-back length (bars), short trades }
        NS4 (50),          { Max look-back length for adaptive indicator (bars),
short trades }
        NS5 (37),          { Indicator look-back length (bars), short trades }
        NS6 (22),          { Indicator look-back length (bars), short trades }
        NS7 (6),           { Indicator look-back length (bars), short trades }
        XS1 (4.5736),      { TrendParam value for adaptive indicator (-5 to 5), short
trades }
        XS2 (2.0417),      { TrendParam value for adaptive indicator (-5 to 5), short
trades }
        XS3 (0.9537),      { Multiple of price difference (e.g., ATR), exit, short
trades }
        NBarExL1 (11),     { Number of bars from entry for market exit if
unprofitable, long trades }
        EntrySzS (790.00), { Value of fixed size entry (stop/limit) per
share/contract, short trades }
        TargFrS (0.9144),  { Multiple of price difference (e.g., ATR), exit, short
trades }
        NBarExS1 (10),    { Number of bars from entry for market exit, short trades }
        PSParam (1.00),    { Position sizing parameter value }
        RoundPS (true),    { Round-to-nearest (true/false) }
        RoundTo (1),       { Round-to position size value }
        MinSize (1),       { Minimum allowable position size }
        SizeLimit (100);   { Maximum allowable position size }

{ Variables for entry and exit prices }
Var:    EntPrS   (0),
        TargPrS  (0);

{ Variables for entry and exit conditions }
Var:    VarL1     (0),
        VarL2     (0),
        VarL3     (0),
        VarS1     (0),
        VarS2     (0),
        VarS3     (0),
        VarS4     (0),
        VarS5     (0),
        VarS6     (0),
        VarS7     (0),
        EntCondL  (false),
        EntCondS  (false),
        ExCondL   (false),
        ExCondS   (false);

{ Variables for position sizing }
Var:    NShares  (0);

{ Entry price }
EntPrS = C - EntrySzS/BigPointValue;

{ Entry and exit conditions }
VarL1 = DayOfWeek(date);
VarL2 = FastD(NL2);
VarL3 = FastK(NL3)[ShiftL1];
VarS1 = AdaptiveIRSI(O, NS1, NS2, XS1);
VarS2 = AdaptiveIRSI(L, NS3, NS4, XS2);
VarS3 = Average(VarS2, NS5);
VarS4 = StandardDev(O, NS6, 1);
VarS5 = VarS4 * XS3;
VarS6 = Momentum(L, NS7);
```

```
VarS7 = AbsValue(VarS6);
EntCondL = VarL1 <= NL1;
EntCondS = VarS1 crosses above VarS3;
ExCondL = VarL2 >= VarL3;
ExCondS = VarS5 >= VarS7;

{ Position sizing calculations }
NShares = PSParam;

If RoundPS and RoundTo > 0 then
    NShares = IntPortion(NShares/RoundTo) * RoundTo;

NShares = MaxList(NShares, MinSize);
NShares = MinList(NShares, SizeLimit);

{ Entry orders }
If MarketPosition = 0 and EntCondL then begin
    Buy("EnMark-L") NShares shares next bar at market;
end;

If MarketPosition = 0 and EntCondS then begin
    Sell short("EnStop-S") NShares shares next bar at EntPrS stop;
end;

{ Exit orders, long trades }
If MarketPosition = 1 then begin
    If ExCondL or (BarsSinceEntry >= NBarExL1 and C < EntryPrice) then
        Sell("ExMark-L") next bar at market;
end;

{ Exit orders, short trades }
If MarketPosition = -1 then begin
    TargPrS = EntryPrice - TargFrS * TrueRange;
    Buy to cover("ExTarg-S") next bar at TargPrS limit;

    If ExCondS or BarsSinceEntry >= NBarExS1 then
        Buy to cover("ExMark-S") next bar at market;
end;
```

For additional examples, please see the Newsletters section on the Adaptrade.com web site.

Chapter 2

Getting Started

Installation

Recommended System Requirements:

3 GHz or faster processor
8 GB or more RAM
64-bit Windows Vista or newer operating system
Monitor: 17 inch or larger

The trial version of Adaptrade Builder can be downloaded from the trial download page on the Adaptrade.com website. The licensed version of Builder can be downloaded by clicking the link in your purchase receipt. The installation file for Builder should look like the following setup files for version 5.0.0: BuilderTrialSetup_v500.exe (trial version) or BuilderLicensedSetup_v500.exe (licensed version). To install the program, browse to the location of the installation file via Windows Explorer and double-click the file to open and run it. Alternatively, select Run from the Accessories menu under the list of programs in the Start menu, browse to the location of the file, click Open, then click OK in the Run window. The installation should begin.

For the licensed version, the installation program will prompt you for a password, which was provided with the installation file. The installer 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.x\, where x.x.x is the version number (e.g., 5.0.0). You can install the program in another location if you wish.

For the licensed version, please be sure to save the installation file and password in a secure location, such as Dropbox, OneDrive or on a thumbdrive, for future installations. Adaptrade Software cannot guarantee that this version of Builder will be available for download in the future.

Note about 32 vs. 64-bit versions: Current versions of Builder are available only for 64-bit versions of Windows, which allows the program 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. **Current versions should run on all recent releases of Windows from Windows 7 to the present. Since Builder is very computationally intensive, older hardware and older versions of Windows, such as XP, are not recommended.**

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. In addition, the bonus strategies and other bonus files should be located in the **Bonus** folder in the Adaptrade Builder folder. The file ReadMe-BuilderBonusFiles.htm in Bonus describes the files in the Bonus folder.

Note: While the licensed version of Builder and the trial version are functionally identical, the **trial version** only runs for the length of the trial period and **cannot be converted to a licensed copy**. The licensed version of Builder must be downloaded and installed after purchase via the Download link provided at that time. The licensed version can be installed over the trial version.

Platform Files: Builder includes several functions and indicators that need to be installed into the trading platforms for Builder-generated strategies to run properly. If these functions are not installed, the strategies generated by the program will not compile or run. These functions must be installed separately for each platform you intend to use for trading. **Please read the sections below for the platforms you intend to use.** To determine if the platform files have been updated for the current release and therefore need to be re-installed (if installed previously), please refer to the Release Notes page at Adaptrade.com or see the notes on the trial download page for the current release.

TradeStation/MultiCharts Installation Notes. Several files that work in conjunction with Builder need to be installed separately into TradeStation or MultiCharts. These files can be found in the EasyLanguage folder in the Adaptrade Builder folder under Program Files after Builder is installed. For current versions of TradeStation, the files end with the file extension .eld. For TS 2000i, the file extension is .els, and for MultiCharts, the extension is .pla. The files should be imported into TradeStation using the import command of the File menu. For MultiCharts, the files can be imported via the Import command of the File menu in the PowerLanguage Editor. Please refer to the platform documentation for instructions on importing .eld/.els/.pla files into TradeStation/MultiCharts.

The specific files that need to be imported into TradeStation/MultiCharts include NNCOMPUTE, which contains functions for use with the neural network indicator option in Builder, and BUILDER_INDICATORS_V3 (e.g., BUILDER_INDICATORS_V3-TS.ELD for current versions of TS), which contains the adaptive indicators (Adaptive VMA, Adaptive ZLT, etc.).

NinjaTrader 7/8 Installation Notes. Strategies created with either the NinjaTrader 7 or NinjaTrader 8 code option require several indicator and strategy files to run. Without these files, the NinjaScript code will not compile or execute. These files are contained in a zip file named NinjaTrader.Adaptrade.zip or NinjaTrader8.Adaptrade.zip for NinjaTrader 7 or NinjaTrader 8, respectively, which can be found in the NinjaScript7 or NinjaScript8 folder in the Adaptrade Builder folder under Program Files after Builder is installed.

To install the files into NinjaTrader 7, open NinjaTrader 7, select Utilities from the File menu and select Import NinjaScript… Browse to the NinjaScript7 folder in the Adaptrade Builder installation folder and select the NinjaTrader.Adaptrade.zip file. Follow the prompts to install the files into NinjaTrader 7.

To install the files into NinjaTrader 8, open NinjaTrader 8, select Import from the Tools menu and select NinjaScript Add-On... Browse to the NinjaScript8 folder in the Adaptrade Builder installation folder and select the NinjaTrader8.Adaptrade.zip file. Follow the prompts to install the files into NinjaTrader 8.

MetaTrader 4 Installation Notes. MetaTrader 4 strategies created by Builder require files contained in three folders: Include, Indicators, and Libraries. These folders can be found in the MT4 folder in the Adaptrade Builder folder under Program Files after Builder is installed. The files in these folders provide basic functionality that any MetaTrader 4 strategy generated by Builder may require. All the files contained in the three folders should be copied to the corresponding folders within your MetaTrader 4 installation so that MetaTrader 4 can find

them when compiling strategies. For MT4 version 600 and newer, the correct folder locations can be found using the "Open Data Folder" command of the File menu in MT4. The general form of the folder path is C:\Users\User_account_name\AppData\Roaming\MetaQuotes\Terminal\Instance_id, in which User_account_name and Instance_id are specific to the user's computer. Within this folder, the Include, Indicators, and Libraries folders are located in the MQL4 folder.

In versions of MT4 prior to version 600, the three folders are located in the installation folder in the experts folder (C:\Program Files (x86)\MetaTrader 4\experts\).

AmiBroker Installation Notes. AmiBroker strategies created by Builder require an associated "include" file: ASBuilderCommon.afl. This file can be found in the AFL folder in the Adaptrade Builder folder after Builder is installed. This file needs to be copied to the Include folder in the AmiBroker Formulas folder under Program Files. The full path under 64-bit Windows is C:\Program Files\AmiBroker\Formulas\Include. This file provides basic functionality that all AmiBroker strategies generated by Builder require.

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

- Since new versions of Builder install in folders that are specific to the version number, it should not be necessary to uninstall a prior version of Builder before installing a newer one. It's not necessary to uninstall the trial version prior to installing the licensed version of Builder.
- Uninstalling an older version will not affect any project 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. Use File -> Save As if you wish to retain both versions.

Builder can be uninstalled through the Windows Control Panel.

Please note that uninstalling and reinstalling Builder will not affect the pane layout and other user interface settings. These settings are stored in the Windows registry. All panes in Builder can be moved and resized by clicking and dragging, as explained in Program Layout and Configuration, below. You can reset the layout by clicking the Reset Layout button on the View menu.

Program Layout and Configuration

The user interface of Adaptrade Builder consists of several different windows and panes. The overall layout of the program can be changed by clicking the Strategy Builder button in the Build menu, which can be seen in the upper left-hand corner of Fig. 1.1. This button switches the layout between charting mode and building mode. In building mode, the program includes all the results windows and panes used to display the results of the build process, as shown in Fig. 1.1. In charting mode, shown below in Fig. 2.1, the charting pane becomes more prominent, and the build-related panes are closed. As an alternative to clicking the Strategy Builder button, any of the different windows and panes can be opened or closed at any time by clicking the corresponding boxes on the View menu.

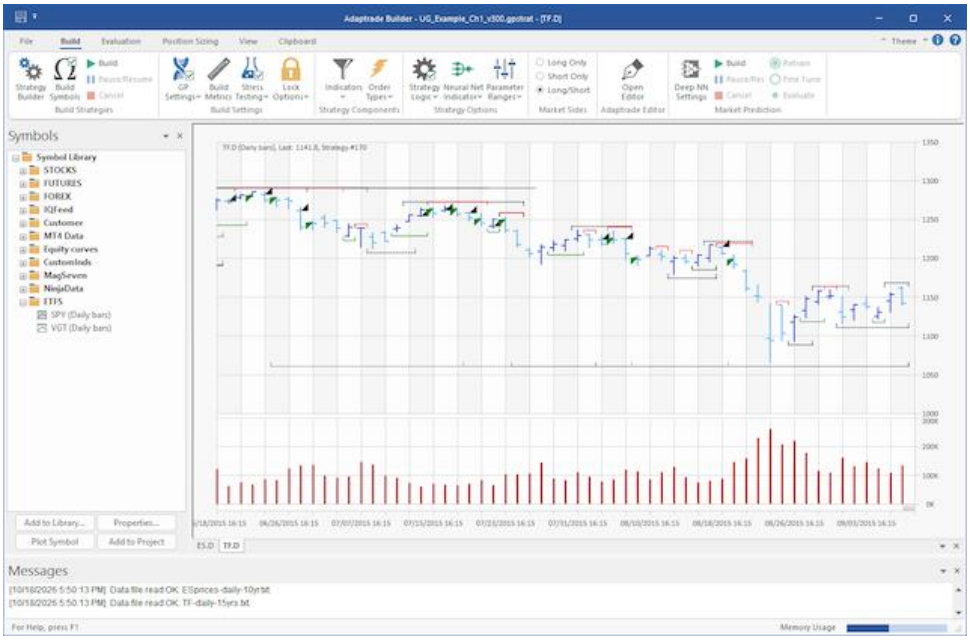


Figure 2.1. Adaptrade Builder in charting mode.

The so-called ribbon menu is located at the top of the main window and consists of the File, Build, Evaluation, Position Sizing, View, and Clipboard menus. In the Build menu, most of the buttons activate drop-down windows for entering build settings, such as the genetic programming (GP) settings, build metrics, strategy logic, and so on. To save space, the ribbon menu can be minimized by clicking the ^ button in the upper right-hand corner of the main window next to the Theme menu.

Aside from the ribbon menu and the chart window, most of the other windows displayed in build mode are referred to as docking windows or panes. Docking panes are floating windows that can be re-positioned by clicking and dragging. The panes shown in Fig. 1.1 consist of the Symbols, Build Results, Messages, and the set of tabbed panes for the Performance Report, Build Report, Equity Curve, Trade List, and Strategy Code windows.

The Symbols pane (i.e., symbol library) displays all the market symbols that can be plotted or used in building strategies. To build a strategy for a new market, the symbol must be added to the symbol library through this pane. Once added to the library, the associated price data are available to any project.

The Build Results pane displays summary statistics for all strategies in the population. This is the main display of results from the build process, where each row in the table represents a strategy in the population. Clicking on a row in the table changes the display of results in the tabbed results windows (Equity Curve, Trade List, Strategy Code, etc.), which display the results for the selected strategy. In addition, if the option to show the strategy trades on the price chart has been selected, the chart(s) will be updated to display the trades for the selected strategy.

There are several non-pane windows as well. The Build Symbols window is where you select the markets to use in the build process. Similarly, the Evaluation Symbols window allows you to select symbols for evaluating strategies separate from the build process. Another non-pane window is the Build Progress window, which displays two graphs that show the progress of the build process while it's ongoing. This window can be left open while working

elsewhere in the program or it can be minimized or closed as desired. It can also be opened or closed at any time through the View menu.

The Trading Orders window is also a non-pane window. This window is displayed when the option to "Show Trading Orders" is selected on the right-click context menu of the chart window. This window can also be kept open while working elsewhere in the program and can be opened and closed through the View menu.

Changing the Pane Layout
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 Equity Curve and Strategy Code windows. The tabs can be moved relative to each other by clicking and dragging.

To rearrange the pane windows, sometimes it's necessary to move them in multiple steps. For example, to switch the locations of the symbols pane and the set of tabbed results panes, you could move the symbol pane to the right of the tabbed results panes first, then move the tabbed results panes as a group to the left of the charts. Because the chart view window is not a docking window, it can't be moved directly. If you want to move the chart windows to a different location, you can move the other panes relative to them. For example, if you want the chart windows 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 chart windows 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.

Deleting the Registry Entry
When Builder is first installed, it arranges the panes in the default configuration shown in Fig. 1.1. To reset the program layout to the default configuration, click the Reset Layout button on the View menu. Uninstalling and reinstalling Builder won't affect the layout because the layout is stored in the registry, which is not deleted during uninstall. If it's necessary to change a setting not affected by the Reset button, the registry entry must be deleted. To do this, use the Windows Regedit program. Type "regedit" into the search box in the Windows Start menu, browse to the location of the registry entry for Builder, and delete the folder SettingsXXX, where XXX is the version number (e.g., Settings500 for version 5.0.0). The registry entry for Builder is located at HKEY_CURRENT_USER\Software\Adaptrade Software\Builder\.

In addition to the pane layout, the registry also stores the name of the project file to open when Builder first starts up. If this file becomes corrupted or damaged for any reason, it may prevent Builder from starting correctly. If you believe this has happened, you can try deleting the registry entry, which will prevent the program from trying to open the damaged project file when it starts up.

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 Build Results tables, along with the performance results for each strategy. Only one project file can be open at a time.

When New is selected from the File menu, the new project is given the default name "BuilderProject" combined with the date and time in YYYYMMDDHHMM format (e.g., BuilderProject202610161234). When Save is subsequently selected, you'll be prompted to enter a new file name. If Close is selected from the File menu, you'll be prompted to save the file if it has not been saved since last changed, and the current project file will be closed. When no project file is open, most of the menu commands are unavailable, and you'll be unable to start a new project until selecting New.

If a project file is open at the time the program is exited, that project file will be automatically opened upon start-up the next time Builder is run. As explained in the previous section, if this file becomes corrupted or damaged for any reason, it may prevent Builder from starting correctly. If you believe this has happened, you can try deleting the project file, renaming it, or deleting the registry entry, which will prevent the program from trying to open the damaged project file when it starts up.

Quick Start Steps

As explained in the introduction, Adaptrade Builder uses an evolutionary algorithm to build populations of trading strategies or populations of deep neural networks (DNNs). This section presents the steps involved in building trading strategies. The process of building populations of DNNs is much the same and was covered in the section Deep Neural Networks.

The process of building trading strategies in Adaptrade Builder begins by specifying the price data or market symbols. Next, a series of settings is entered via the ribbon menu. The settings determine the types of strategies that are built and control various aspects of the build process. While the build process is ongoing, progress messages are displayed in the Messages window. Additionally, the results windows (Build Results, Performance Report, Build Report, Equity Curve, Trade List, and Strategy Code) are updated after each generation is completed.

Step 1. Obtain the Price Data

Builder is designed to read files of price data saved in text 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. 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. In MultiCharts, select "Export Data" from the File menu to save the chart data to a text file. In MetaTrader 4, go to the Tools menu and select History Center. Select the symbol from the list at left and click the Export button. In NinjaTrader 7, use the Export tab on the Historical Data Manager window. In NinjaTrader 8, select Export → Historical Data from the Tools menu. For AmiBroker, the price data should be exported to a text file using the software provided by your data vendor.

Step 2. Add the Price Data File Obtained in Step 1

Open Builder and, in the Symbols pane, select a folder, right-click, and select the Add New Symbol command or click the Add New Symbol button at the bottom of the Symbols pane. This will open a "Select Symbol File" dialog window that will enable you to select the text

file of price data obtained in Step 1. Select the file of price data and click the Open button. The Data File Format window, as shown in Fig. 2.2, will then open.

The top table displays the contents of the price file and enables you to set the formatting that tells the program how to read and interpret the data in the file. Select the field delimiter (comma, semi-colon, tab, or space) used in your file to separate the data fields. The default column headings for the data fields are date, time, open, high, low, close, and volume. Click any column heading in the table to change, add, or remove its label. Each column of data you want the program to read should have a label. **Note that the labels, if any, that are part of your data file don't determine the file formatting -- you need to ensure that the label of each column heading matches the format of the data in the corresponding column.** 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. Click the Clear Column Headings button to remove all column labels to start fresh.

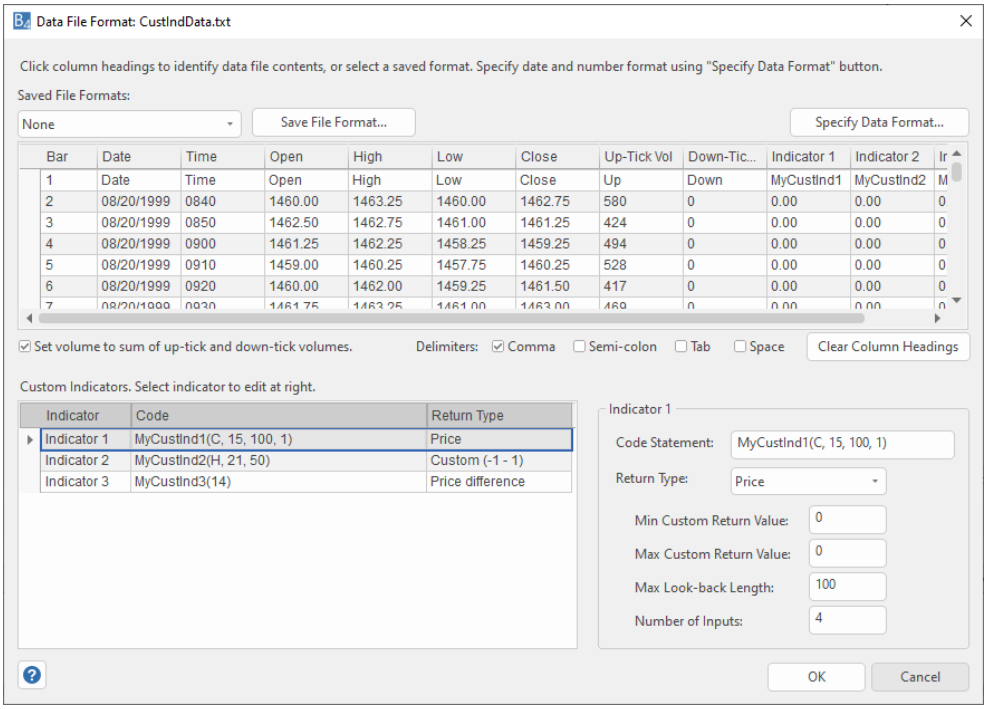


Figure 2.2. The format of the price file is selected on the Data File Format window.

Click the Specify Data Format button to specify how the date and price fields are formatted. This will open the Data Format window (see the Symbols and Settings chapter), which allows you to select either a locale for reading the data, such as "English (US)", or a set of specific formatting options, such as MMDDYYYY for the date format. Data formats can be saved for future use. Similarly, the file formatting options selected on the Data File Format window, including the data format, can be saved for future use, as indicated by the Save File Format button in Fig. 2.2.

Once you've selected the format for the file of price data, clicking OK will open the Symbol Properties window, shown below in Fig. 2.3. Enter values for each entry in the list of properties, including symbol, bar type, point value, and so on. 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. Click OK to return to the Symbols pane.

To add the newly entered symbol to the build process, where it will be used to build your strategy, right-click on the symbol and select Add to Project or click the Add to Project button at the bottom of the Symbols pane. Any symbols added in this manner will be displayed in the Market Symbols table in the Build Symbols window, shown below in Fig. 2.4, as well as in the Evaluation Symbols window.

Click the Add button on the Build Symbols window to add a price file from the symbol library to the Market Symbols table or click Delete to delete a symbol from the table. Select the files to include in the build by checking the boxes in the Include column. Select the data series for each symbol using the Series column. The symbol or symbols on which the trades are placed are the "Primary" series. If more than one primary series is selected, the strategies will be evaluated on all such symbols with trades placed on all primary symbols. The reported performance will represent the combined results from all primary markets. Symbols marked as 2, 3, 4, and so on, are secondary series. The program may incorporate these symbols into the strategy logic depending on the build process. In TradeStation, for example, these series are referred to as "data2", "data3", "data4", etc.

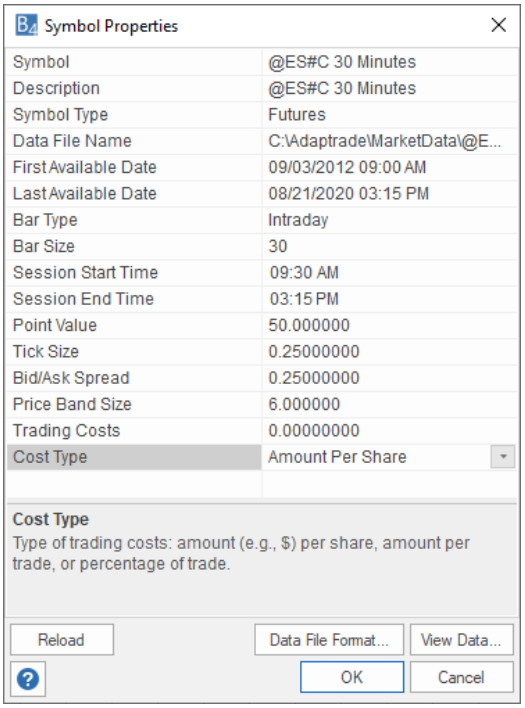


Figure 2.3. Symbol Properties window.

The settings in the Market Symbols table for Bar Type, Bar Size, and Trading Costs are the settings you select for building and/or evaluating a strategy. These can be different from the values for the symbol in the library. For example, if the price file consists of 1 minute bars, the settings in the Symbol Properties window should always specify 1 minute bars, but once added to the Build Symbols window, you can change the bar type to any multiple of 1 minute or to daily, weekly, monthly, or range bars.

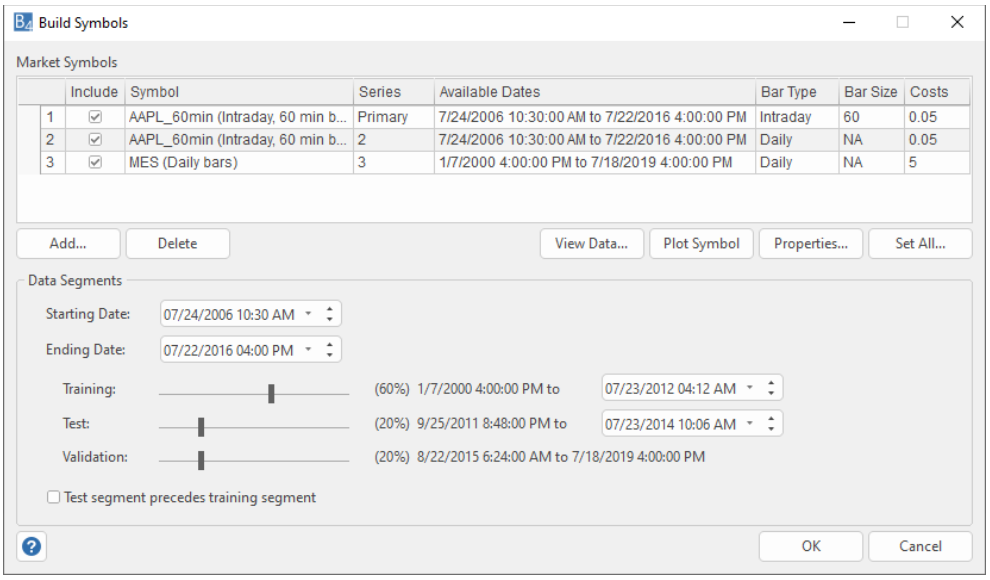


Figure 2.4. Settings for the markets used in the build are made on the Build Symbols window.

Step 3. Select the Date Ranges

The Build Symbols window also includes controls for changing the start and end dates for analysis as well as the training/test/validation ranges. You can change the start and end dates for building a strategy by clicking on the calendar controls for "Starting Date" and "Ending Date". Move the sliders to change the amount of data used for the training, test, and validation segments. The training data are used in building the strategy. Test data are used in evaluating the results on the training segment, which helps to guide the build process. The validation data are used to verify the final results. Because the validation data are not used during the build process, the results achieved on validation data are unbiased. The default percentages are 60/20/20 for training/test/validation.

Step 4. Choose the Genetic Programming (GP) Settings

The GP Settings button in the Build Settings panel of the Build menu opens the GP Settings drop-down window, as shown in Fig. 2.5. Different options that influence the build process are selected on this window. For example, this is where you enter the population size and the number of generations. You can also select from two different Build Failure Rules, which will restart the build process after a specified number of generations if the results on the test segment are not satisfactory. The build process can be terminated (Build Termination Options) when a maximum number of generations has been completed or when the fitness on the test segment starts to decline.

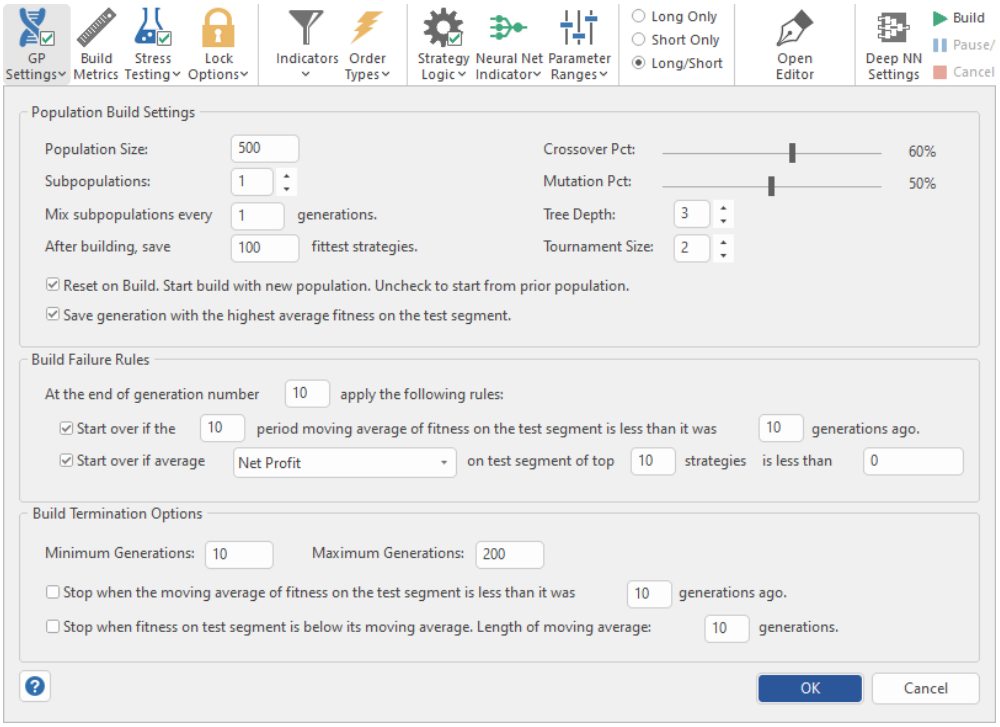


Figure 2.5. Options that affect the build process are specified on the GP Settings drop-down window.

You can also modify the crossover and mutation percentages, tree depth, and tournament size on this window. 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 "After building, save [] fittest strategies" option. If the Reset on Build box is checked, the population will be initialized randomly, which is the default.

Step 5. Select the Build Metrics

The Build Metrics window is shown in Fig. 2.6. This is where you select the build objectives and conditions that will guide the build process. Strategies are ranked during the genetic programming process according to the *fitness*, which is a combination of the build objectives and conditions selected on this tab. The objectives are metrics that you want to minimize (e.g., drawdown) or maximize (e.g., net profit). The conditions are expressed as inequality or equality statements, such as a correlation coefficient greater than or equal to 0.95 or the number of trades between 100 and 300. The third list box is for selecting conditions for filtering the strategies for the Top Strategies results. Use the buttons adjacent to each list to add, modify, or remove entries from the list. A right-click (context) menu is also available, and items can be cut (Ctrl-X), copied (Ctrl-C), and pasted (Ctrl-V).

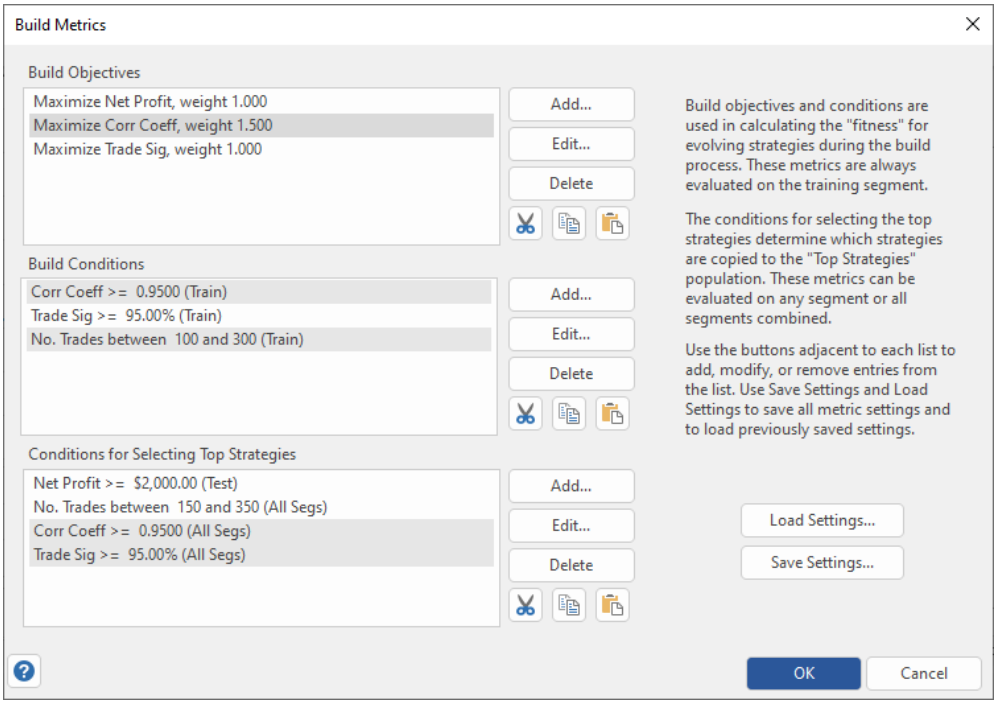


Figure 2.6. Build metrics are specified in the Build Metrics window by selecting objectives and conditions for key performance metrics.

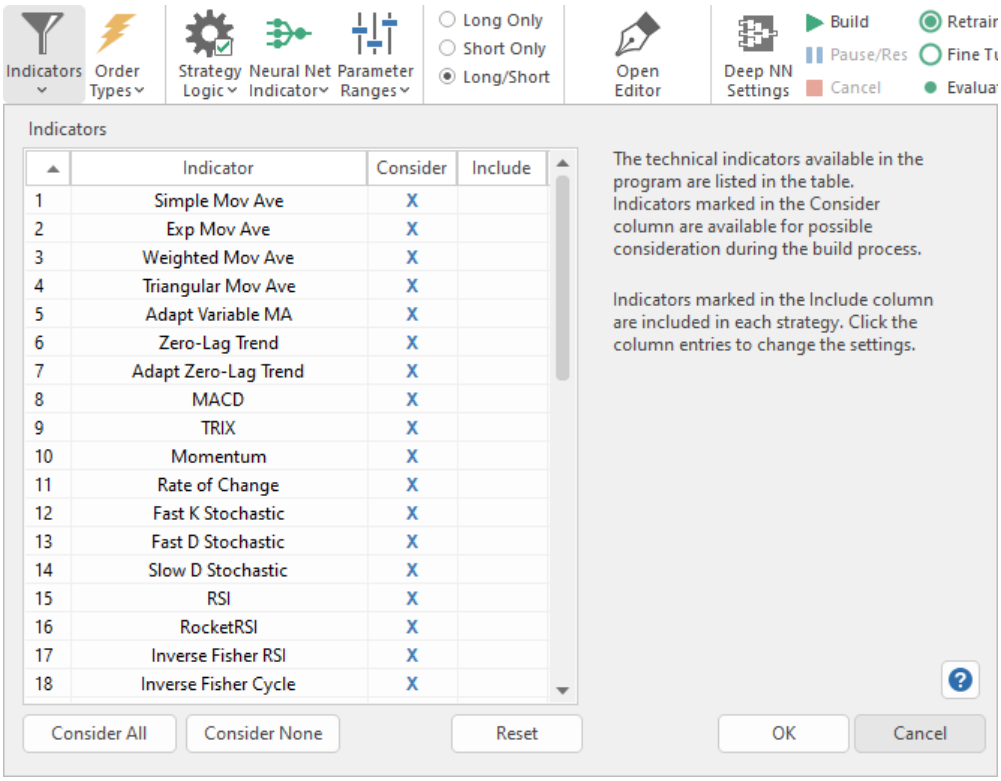


Figure 2.7. The indicators available in Builder are listed on the Indicators window. Indicators can be removed from the build set by un-checking the item in the Consider column and added to each strategy using the Include column.

Step 6. Select Indicators and Order Types

Indicators are selected on the Build menu by clicking the Indicators button, which opens the Indicators window, shown in Fig. 2.7. The available order types are listed in the table on the Order Types window, shown in Fig. 2.8. The tables on these two windows represent the build set, which contains the list of 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 row in the Consider 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 indicator or order type in the Include column instructs the program to include the item in each generated strategy.

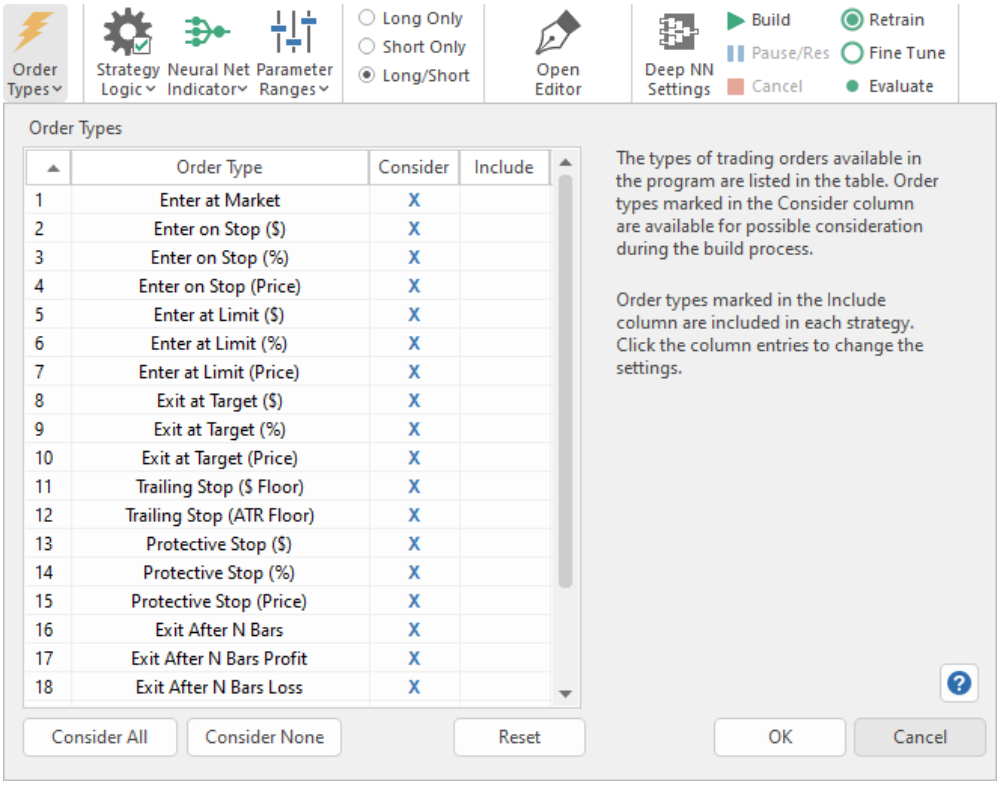


Figure 2.8. The types of trading orders available in Builder are listed on the Order Types window. Order types can be added or removed from the build set using the Consider column and added to each strategy using the Include column.

Step 7. Select Strategy Options on the Build Menu

The Build menu is shown in Fig. 2.9. The settings in the Strategy Options panel control the types of strategies generated by Builder. Clicking the Strategy Logic button opens a drop-down window containing options for long/short symmetry, trade entry and exit time ranges, a limit on the number of entries per day, and other strategy logic options. The Neural Net Indicator drop-down window allows you to add a neural network indicator to the strategy and set the associated parameters. Clicking the Parameter Ranges button opens a drop-down window that allows you to enter 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. Also shown in Fig. 2.9 is the Market Sides panel, which allows you to restrict entries to either long or short trades only or to include both long and short trades in the same strategy.

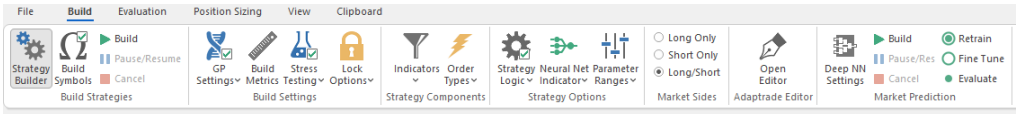


Figure 2.9. The Build menu provides access to most of the settings related to building strategies.

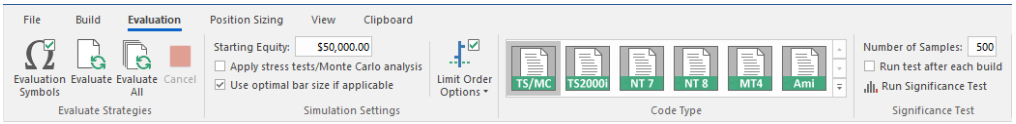


Figure 2.10. Options that affect how strategies are evaluated are set on the Evaluation menu.

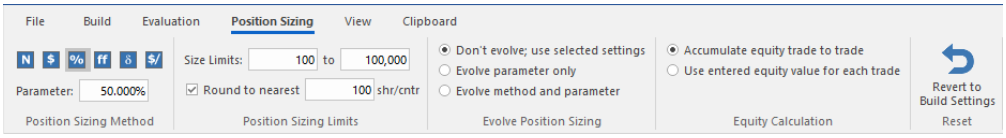


Figure 2.11. Position sizing options determine how many shares or contracts are traded.

Step 8. Select Evaluation and Position Sizing Options

Options that affect how strategies are evaluated are selected on the Evaluation menu, shown in Fig. 2.10. The principal options on this menu are the starting equity value, which can affect position sizing, and the code type. Position sizing settings are made on the Position Sizing menu, shown in Fig. 2.11. You can choose from six different position sizing methods. You can also choose whether you want the build process to select the method for you and/or whether the build process will select the position sizing parameter value, such as the fixed fraction for fixed fractional position sizing. Other options on this tab include the maximum number of shares or contracts that will be traded and whether to round the position size to the nearest value, such as rounding the number of shares to the nearest 100.

Select the code type for your target trading platform prior to building. While it's possible to change to a different code type after building, different results may be found because some indicators are evaluated differently for different platforms and some indicators and strategy options are not available for some code types.

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 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 9. Start the Build Process

Click the Build button in the Build Strategies panel of the Build menu (Fig. 2.9) to start the build process. A message is displayed in the Messages window near the bottom of the main window during each step of the process. After each generation is completed, the results windows (Build Results, Performance Report, Build Report, Equity Curve, Trade List, and Strategy Code) are updated. The summary performance metrics for each of the strategies are displayed in the Build Results tables. Click on a row in the results table to display the results for the corresponding strategy in the results windows. The build process can be cancelled at any time by clicking the Cancel button or paused by clicking the Pause/Resume button.

Step 10. Evaluate the Generated Strategies and Adjust if Necessary
After the build termination conditions are met or the Cancel button is clicked, the build process stops. The generated strategies are listed in the Build Results tables in order of decreasing fitness. Builder will save any number of strategies you specify on the GP Settings window up to the population size. Evaluate the saved strategies by reviewing the build and performance reports, equity curve, trade entry and exit points on the price chart, and trade-by-trade results in the Trade List table. If none of the saved strategies meet your requirements, change the build objectives and conditions on the Build Metrics window 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 11. Transfer the Code to Your Trading Platform
When you're satisfied with the results in Builder, transfer the strategy code to your trading platform for any additional testing, such as real-time tracking, and for trading. The code in the Strategy Code window can be copied by right-clicking and selecting "Copy Strategy". In TradeStation or MultiCharts, open a new strategy window in the EasyLanguage editor, choose a name, then paste the code into the empty strategy window. Finally, compile the EasyLanguage code.

In NinjaTrader 7/8, right-click on the Strategy Code pane and select "Save NinjaScript Strategy to File". Choose a name for the strategy or accept the default and save the strategy to the folder Documents\NinjaTrader 7\bin\Custom\Strategy\ (for NT7) or Documents\NinjaTrader 8\bin\Custom\Strategies\ (for NT8). Open the strategy in the NinjaTrader 7 editor by selecting Edit NinjaScript from the Tools menu and click the Compile button. In NinjaTrader 8, open the editor via the File menu. The strategy should then be ready to back-test via the Strategy Analyzer.

In MetaTrader 4, open a new window in the MetaEditor, paste in the code from Builder, and click the compile button. Alternatively, you can save the MetaTrader 4 code directly to a .mq4 file by right-clicking in the code window in Builder and selecting "Save MT4 Strategy to File".

In AmiBroker, in the Charts pane, right-click on the Systems folder and select "New" and then "Formula". Enter a name then right-click on the name and select "Edit". Select "Paste" from the Edit menu to paste the code into the edit window then click the save icon and, finally, click the AFL icon to verify the code.

To test the strategy in TradeStation/MultiCharts, 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 Build Results table.

In MultiCharts, the value is entered under Format Signal, Properties ("Maximum number of bars study will reference"). This ensures that the strategy has enough data to start calculating the indicators at the beginning of the chart.

In NinjaTrader 7/8, this value is set by the strategy code and can be seen in the Strategy Analyzer window ("Min. bars required" in NT7 or "Bars required to trade" in NT8).

In MetaTrader 4, select the strategy (Expert Advisor) on the Tester window (Settings tab) and select the symbol and date range. For best results in MetaTrader, select "Every tick" as the Model then click the Start button to run the back-test.

In AmiBroker, right-click on the strategy in the Systems folder of the Charts pane and select "Analysis". On the Analysis window, reset the parameter values by clicking on the Parameters icon and check the settings by clicking on the Settings icon. Finally, click the Backtest icon to run the test.

Note: *If you're pasting the strategy code over the code for an existing strategy, it will be necessary to reset the input values before running the back-test. In TradeStation, delete the strategy from the chart and re-insert it to reset the input values. In MetaTrader 4, this can be done by clicking the Expert properties button on the Tester window and clicking the Reset button on the Inputs tab. In AmiBroker, click the Parameters icon on the Analysis window and click "Reset all".*

To ensure that you get the same results in the trading platform as in Builder, make sure that the basic settings, such as trading costs and point value, are set correctly. For example, in TradeStation, costs should be 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 in TradeStation/MultiCharts is the same. Small differences may still occur, due primarily to slight calculation differences. Because the series of trades in a back-test are rarely independent from one another, when one trade exits at a different bar, it can cause subsequent trades to enter and exit at different prices as well. See the section Common Questions (Usage Topics) for more information on possible sources of difference between the results in Builder and those in the trading platform.

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

Symbols and Settings

Symbol Library

The Symbols pane, as shown in Fig. 3.1, contains the symbol library. In Builder, symbols can include both price data symbols and symbols representing external equity curves. The latter can be used to build strategies that work optimally with an existing market system or portfolio, as represented by its equity curve. Both types of symbols are added to the program in the same way. Equity curve symbols are discussed further in Equity Curve Symbols, below.

Before market symbols can be added to the Build Symbols window for building strategies, they must be added to the symbol library through the Symbols pane. The Symbols pane organizes all the symbols in Builder in a series of folders arranged in a tree structure, like the file system in Windows Explorer.

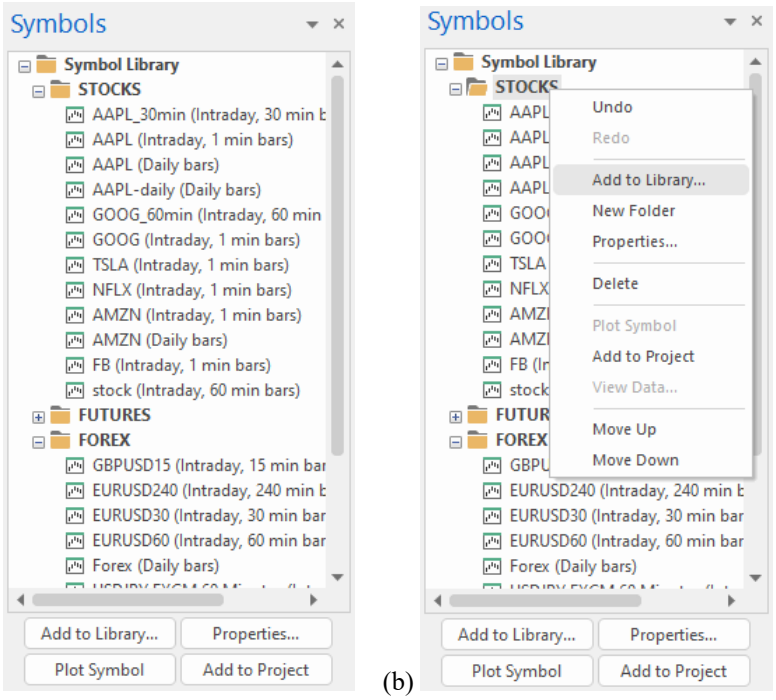


Figure 3.1. (a) The Symbols pane contains the symbol library. (b) The right-click (context) menu contains all commands that apply to symbols and folders.

The symbol library is stored in a separate file ("AB_SymbolLibrary.symlib") and is independent of the project files. The symbol library is accessible whenever the program is running, so that any symbol added to the library is available to any project. A back-up copy of the symbol library file ("AB_SymbolLibrary_backup1.symlib", "AB_SymbolLibrary_backup2.symlib", up to " AB_SymbolLibrary_backup10.symlib") is automatically made at program start-up if the same file has not already been backed up.

Under current versions of Windows, the symbol library and backup copies are stored in the folder C:\Users\user_name\AppData\Roaming\AB_SymbolLib_v30\, where user_name is your user name. Under Windows XP, the path is C:\Documents and Settings\user_name\Application Data\ AB_SymbolLib_v30\. The symbol library is saved automatically whenever a change is made to it. Under normal circumstances, it should not be necessary to use the backup copies. The Undo command (see below) can be used to revert to a prior state if changes are made to the library unintentionally, such as accidentally deleting a symbol that you want to keep. However, commands cannot be undone once the program is closed. For example, if you were to delete a symbol, close Builder, and later decided you want to keep the deleted symbol, you could replace AB_SymbolLibrary.symlib with one of the backup copies.

Any number of folders can be added below the root-level "Symbol Library" folder, which cannot be changed. In each of the folders within the Symbol Library folder, any number of sub-folders can be added. The default configuration includes folders named "STOCKS", "FUTURES", and "FOREX". The name of any folder, including the default folders, can be changed by clicking on the name twice or by clicking the folder once and selecting "Properties" from the right-click (context) menu, shown above in Fig. 3.1b.

To add a folder, click the item below which you want to add the new folder and select "New Folder" from the context menu. New symbols are added similarly by selecting a symbol or folder in the tree and selecting "Add to Library" from the context menu. The new symbol defaults to the properties of the selected symbol, if any. Double-clicking an existing symbol or selecting "Properties" for a selected symbol opens the Symbol Properties window, shown previously in Fig. 2.3 in Quick Start Steps and in Symbol Settings, below. Adding a new symbol is discussed below. Symbols can be added to any folder, including the root folder.

Symbols can be moved from folder to folder by clicking and dragging or by using the Move Up and Move Down commands of the context menu. Undo and Redo commands allow any command from the context menu to be reversed. Up to 10 levels of undo are permitted.

Other commands for working with the symbol library include deleting a symbol (Delete or Del key), plotting a symbol in a price chart (Plot Symbol), viewing the price data in a table (View Data), and adding the selected symbol to the list of symbols in the Build Symbols and Evaluation Symbols windows (Add to Project). Note that using Add to Project adds the selected symbol to both the Build Symbols and Evaluation Symbols windows, whereas using the Add button on either the Build Symbols or Evaluation Symbols window only adds the symbol to that one window.

The buttons at the bottom of the Symbols pane (Add to Library, Properties, Plot Symbol, Add to Project) provide one-click access to some of the more commonly used commands of the context menu.

Note to users of prior versions of Builder: When the program starts up, it checks the project file being opened to see if it contains a price data file that is not in the library. If it does, the program will ask if you want to add the symbol to the library. This is designed to make it easier to populate the symbol library from existing project files.

Obtaining Price Data

Builder can utilize price data obtained from any source provided the file is in a delimited text format and can be formatted under the Data File Format window, discussed below.

TradeStation
For TradeStation users, the easiest way to obtain the price data file for use in Builder is to 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.

MultiCharts
In MultiCharts, select "Export Data" from the File menu to save the chart data to a text file.

NinjaTrader 7/8
In NinjaTrader, use the Export tab on the Historical Data Manager window to save the data to a text file. In NinjaTrader 8, select Export -> Historical Data from the Tools menu. The resulting file will contain 1-minute bars in semi-colon delimited format. Check the semi-colon delimiter box on the Data File Format window after selecting the file in Builder. Once in Builder, the bar type and size can be changed in the Market Symbols table of the Build Symbols window, as explained in Build Symbols, below.

MetaTrader 4
In MetaTrader 4, go to the Tools menu and select History Center. Select the symbol from the list at left and click the Export button. On the Data File Format window, click the Specify Data Format button to select the formatting options for the date and number formats. Once selected, it may be saved for reuse in the future.

Third-Party Data Providers. If you obtain your price data from a data vendor, the vendor should have a software program that can be used to export the data to different formats and bar sizes. Builder requires data in a text file format, such as csv or txt files. Once in Builder, the bar type and size can be changed in the Market Symbols table of the Build Symbols window, as explained in Build Symbols, below.

Equity Curve Symbols

In addition to price data, you can define symbols representing externally supplied equity curves. If this special type of symbol is included as one of the build symbols, the program will add the equity changes calculated from the equity curve data to the bar-by-bar equity values from the strategy to generate a composite equity curve. The performance metrics are then based on the composite curve. If, for example, the external equity curve represents an existing portfolio, Builder will produce strategies based on the combined performance of the existing portfolio and the generated strategies. The resulting strategies will tend to be optimal in combination with the supplied equity curve. This provides a way to build strategies that maximize the portfolio results.

A few points regarding equity curve symbols are worth noting:

- The data don't have to represent bars or match the interval size of the price data being used. For example, if you're building on five-minute bars, it's not necessary to specify an equity value for every five-minute bar. However, for the highest accuracy, an equity value should be specified for each date/time for which there's a new equity value.
- The only requirement is that there be a date/time value and an equity value for each row of data in the file. The "close" field should be selected on the Data File Format window to represent the equity values. Optionally, you can also specify the high and low values of equity at the specified date/time by specifying columns as the "high" and "low".

- The date range should overlap the date range of the selected price data symbols when building, although the dates don't have to overlap exactly. The equity changes from the equity curve symbol will be added to the calculated equity curve for the strategy where the date/times overlap.
- Multiple overlapping or consecutive equity curve symbols can be included in the same build. The program will combine them based on their equity changes and date/time values.
- The equity curve pane will automatically display the detailed equity curve when building with an equity curve symbol. The detailed equity curve is based on the bar-to-bar equity values and plots the equity versus date, rather than trade number. If the display is changed to the closed-trade equity curve versus trade number, the curve will be based solely on the strategy trades and will not include the external equity curve(s). See the section Equity Curve Window in Build Results.
- When building with an equity curve symbol, it's recommended that the correlation coefficient be included as one of the build metrics. The correlation coefficient is based on the detailed equity curve, which includes the external equity curve(s). Maximizing the correlation coefficient in this case will tend to result in strategies that combine well with the external equity curve(s) so that the portfolio results are optimal.
- Equity curve symbols can be evaluated by themselves. If one or more equity curve symbols are the only symbols selected on the Evaluation Symbols window (i.e., no price data symbols are selected), clicking the Evaluate button will compute the metrics that are obtained from the equity curve, including the total net profit, worst-case drawdown, annual return, and so on. If multiple equity curve symbols are included, the metrics will be calculated from the composite equity curve for all symbols. All metrics based on trade results, such as the average trade and profit factor, will be zero or undefined when equity curve symbols are evaluated by themselves. It's not possible to build trading strategies based solely on external equity curves because without price data the results would be the same for every strategy.

Building a Strategy to Trade with an Existing Portfolio or Market System

The following steps can be taken to build a strategy that will be optimally diversified with respect to an existing portfolio or market system:

1. Save the equity curve for the existing portfolio to a text file. The file should contain a list of dates and equity values, one per line. If the portfolio or market system is available in Builder, you can obtain the equity curve file by right-clicking the equity plot and selecting "Save to Text File".
2. Add the equity curve saved in step 1 to the symbol library as an equity curve symbol. This is done by using the "Add to Library" button, selecting the equity curve file, and setting the symbol type to "Equity Curve" on the Symbol Properties window. The equity values should be assigned to the "close" field on the Data File Format window.
3. On the Build Metrics window, add the correlation coefficient as a build metric in addition to any other metrics, such as net profit, that you would normally use. This will ensure that the equity curve for the portfolio, combining the equity curve for the strategy being built with the external equity curve symbol, will be as straight and uniform as possible. The build process will optimize the strategies for the portfolio equity curve, which will include the external equity curve symbol. The resulting strategies will therefore be optimal when traded in combination with the external equity curve.

For more information, please refer to the article http://www.adaptrade.com/Newsletter/NL-OptimalStratDiverse.htm. (Note: This article was written before the equity curve symbol feature was added to Builder, so it uses a slightly different approach, but the basic concepts are the same.)

Adding a Symbol

As explained above, symbols are added to Builder through the symbol library. Clicking the Add to Library button on the Symbols pane opens the "Select Symbol File" file selection window. This is a standard Windows file selection dialog that allows you to select a text file of price data or data for an external equity curve. Each line in the file should specify the data for a separate date/time, which should be ordered consecutively. Click the Open button to bring up the Data File Format window (see below), which allows you to specify the formatting for the selected file.

After entering the formatting and clicking Ok, the Symbol Properties window will be opened, as shown in Symbol Settings, below. This is where all the properties describing the symbol are entered. Click the name of each property to display a description at the bottom of the properties window. For price symbols, data for the following properties should be entered: Symbol (i.e., name or label for symbol), Bar Type, Bar Size (if applicable), Session Start Time, Session End Time, and Point Value. Each of the symbol properties is described in detail in Symbol Settings. The symbol and description fields default to the file name without the file extension. If the symbol represents an external equity curve, set the Symbol Type property to "Equity Curve". This will limit the number of properties to the ones needed to specify an equity curve symbol.

To change the text file associated with the symbol, click the "..." button in the Data File Name property. This will open the "Select Symbol File" dialog window, which will allow you to select a new file of data. To change the formatting of the data file, click the "Data File Format" button, which will open the Data File Format window.

Data File Format Window

The Data File Format window is used to tell Builder how to read and interpret the data file. 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). If the date and time are together in one field (e.g., 11/5/2012 13:15 or 20090531 13:15), the field should be labeled "date" and no "time" field should be present.

Notice that the first line in the table depicted in Fig. 3.2 contains labels for the data columns in addition to the labels in the column headers. While the program will attempt to use the labels contained in the data file, it's the column headings that define the file formatting, so it's necessary to verify that each column heading has a label that matches the data in the corresponding column.

For an external equity curve, each line in the file should consist of the date and time and the corresponding equity value. The equity value should be labeled as the “close”. Optionally, the high and low equity values may be included on each line by labeling them as the “high” and “low” prices. The high and low equity values are used in calculations, such as for the maximum adverse excursion and drawdown, that are based on intra-bar equity. If the high and low equity values are not provided, the closing equity value is used instead. Any column labeled with a heading other than the date, time, close, high, or low will be ignored for an external equity curve.

Builder is designed to read data files in several text file formats. An example of this data format, as obtained from the TradeStation Data Window, is shown below.

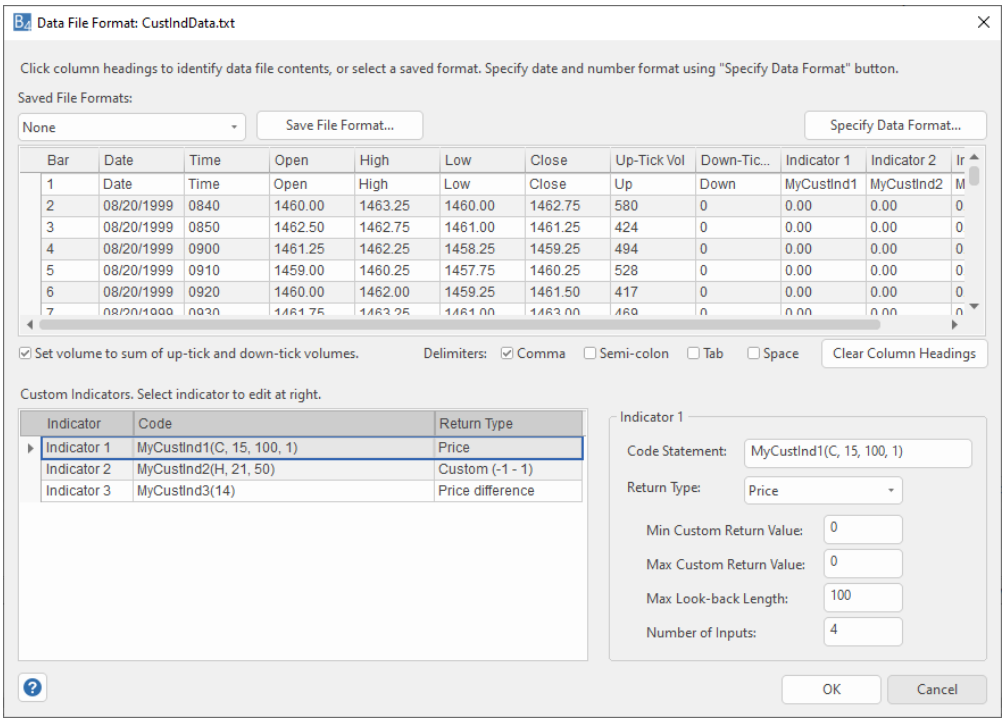


Figure 3.2. The Data File Format window allows you to specify the format of the file of price data.

"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

These data are stored in comma-delimited format, in which commas separate the different fields (date, time, open, high, low, etc.). Other field delimiters can be selected by checking the boxes below the table. Builder can read files delimited by commas, semi-colons, tabs, or spaces. When a new delimiter is selected, the table will be redrawn. If you're not sure how your data file is delimited, try the different delimiters until the fields are displayed in separate columns.

The “Up” and “Down” fields are the up and down tick data for intraday bars. For other bar types, a single volume field will be present. Builder does not use open interest.

The program will attempt to read the column labels from the first line of the file if labels are present, as in the example above. If column labels have not already been selected, the labels read from the first line in the file will be used. The column labels in the first line can be delimited by quotes (""), angular brackets (<>), or left undelimited. If the name of a custom indicator is used as the column label, it will be copied to the table of custom indicator names (see Custom Indicators, below). This can be a convenient way to specify the names of custom indicator functions when using a large number of custom indicators.

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. Select "Indicator" for a column of custom indicator values.

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. Usually, the two fields should be combined for volume-based indicators to evaluate properly. If it appears that a volume-based indicator is not evaluating properly, try deselecting this option.

Clear Column Headings Button
The Clear Column Headings button removes all column headings from the table of price data and removes all custom indicator specifications from the table of custom indicators. This button can be used to initially clear the default column formatting prior to setting up the column headings and custom indicator settings. If you inadvertently remove key settings, such as custom indicator details, click the Cancel button to close the window without saving any changes.

Save File Format Button
Once you've selected the formatting, such as the field delimiter and the column labels, you can save your selections for later reuse by clicking the Save File Format button. This will save your settings in a file, which will then be available for future projects via the drop-down menu labeled "Saved File Formats". The saved formatting settings include those for the custom indicators and data formats, described below. If you have previously saved the file format, you can select it from the drop-down menu. Selecting the option "None" on the drop-down menu will revert the formatting back to the original file format, so *if a new format has been selected or the column formatting has been cleared, selecting "None" from the drop-down will restore the formatting*.

Specify Data Format Button/Data Format window
By default, Builder reads the data file by assuming the data follow the date and number conventions of the "English (United States)" locale, which means dates are assumed to be given as MM/DD/YYYY, prices use a comma as the digit grouping symbol and a period as the decimal symbol (e.g., 1,000.01), among other conventions. If the data file employs other conventions, different formatting options can be selected by clicking the Specify Data Format button, which opens the Data Format window, shown below in Fig. 3.3.

First select either "Use locale" or "Use custom format". For example, to accommodate dates in the format DD/MM/YYYY with numbers in US English format, you can select "Use locale" with the selection changed to "English (United Kingdom)". For more complicated formats, such as the ones often found when saving price data from MetaTrader 4, select "Use custom format".

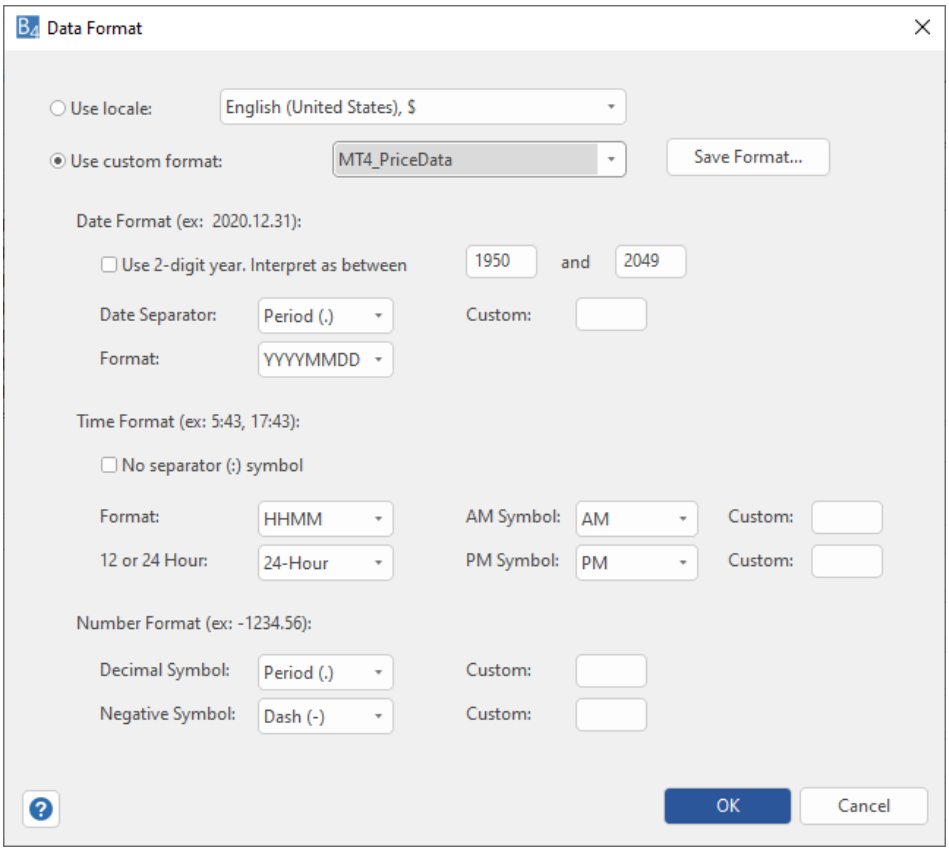


Figure 3.3. The Data Format window allows you to specify the format for reading dates and numbers in the data file.

The custom format selections consist of three sections: date format, time format, and number format. Select the options from those available in the drop-down menus. For example, the date separator can be a slash (/), period (.), dash (-), space, none, or custom. If custom is chosen, enter the custom date separator in the corresponding box labeled "Custom". When a new selection is made, the examples provided in the labels for each section change to reflect the new selection. For example, if "period (.)" is chosen as the date separator, the date example will change to "12.31.2020".

After selecting all the options in all three sections to match the data in your file, you can optionally save the formatting by clicking the Save Format button. This will save the formatting options in a file, which can be accessed via the drop-down menu next to the Save Format button the next time you open the Data Format window.

To return to the Data File Format window, click the OK button to keep your selections or the Cancel button to discard them.

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. In TradeStation and MultiCharts, this can be done by plotting the indicator on the price chart 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 table of the Data 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. 3.2. This also adds an entry to the Custom Indicators table, located below the table of price data in the Data 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 when the strategy is run, and (2) the **return type** of the indicator. For example, in Fig. 3.2, indicator #1 uses the code statement "MyCustInd1(C, 15, 100, 1)." This is the 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 MyCustInd1(C, 15, 100, 1) in a strategy should produce the values shown in the price file under the column labeled Indicator 1.

Note: For MT4, when a custom indicator is added to the strategy code, it has the form iCustom(...., "name", input1, input2, ..., 0, 1). The input values will be copied to the iCustom function from the text between the parentheses in the code statement; e.g., "C, 15, 100, 1" for Indicator 1 in Fig. 3.2. However, any price inputs should be specified as PRICE_OPEN, PRICE_HIGH, PRICE_LOW, and PRICE_CLOSE, in place of O, H, L, and C, respectively. For example, if Indicator 1 above was intended for MT4, it should be written as "MyCustInd(PRICE_CLOSE, 15, 100, 1)".

Note: If a custom indicator cannot be represented in the target language 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. 3.2 is "Price". This means the values produced by the indicator are equivalent to the symbol's price, such as the close, open, high, etc. 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. 3.2. 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 none of the built-in types apply because this will limit the ways in which the custom indicator can be used in the logical conditions*.

Note for True/False Return Types: Custom indicators returning a true/false value should return 0 for false and 1 for true. To convert these values to logical true/false, Builder compares the value to 1. For example, in NinjaScript, MQL4, or AFL, a true/false input value would be converted to true/false as follows: N1 == 1 (in EasyLanguage, it would be N1 = 1). So, if N1 is 0, the preceding statement would return false, and if N1 is 1, it would return true. The same approach applies to variables assigned to true/false custom indicators; e.g., VarL1 == 1 means "VarL1 is true" and VarL1 != 1 means "VarL1 is false", in which VarL1 is the custom true/false indicator. This implies you might see statements such as CondL1 && (VarL1 == 1), CondL1 = VarL2 != N1, CondS1 == (VarS2 != 1), (VarL4 != 1) || (VarL5 == 1), and so on. Currently, no built-in indicators in Builder return true/false, so this only applies to custom indicators returning true/false.

The maximum look-back length required to evaluate the custom indicator can be entered below the fields for the custom return values. For example, in Fig. 3.2, the first custom indicator looks back a maximum of 100 bars. This is the number of bars of data needed to evaluate the indicator and should be entered as the Max Look-back Length. The look-back lengths are used to calculate the Max Bars Back value, which is needed when inserting the strategy into a chart in the trading platform. Similarly, you can enter the number of inputs that the custom indicator uses, which is used in the calculation of the complexity metric.

Including Other Symbols Using Custom Indicators
In prior versions of Builder, if you wanted to include other price series in the build process, such as using 10-year note prices when building a strategy for the E-mini S&P, the other price series needed to be defined as a custom indicator with a custom return type. This enabled statements such as XAverage(ZLTrend((Close of data2), NL3), NL4) <= Average((High of data2), NL1), in which "Close of data2" and "High of data2" were custom indicators. Starting in version 2.5, this is no longer necessary because Builder handles secondary data series natively as part of the build process; see Build and Evaluation Symbols, below.

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.

Note: If multiple markets (price files) are selected as primary data series in the Market Data table on the Build Symbols window and one file contains custom indicators, then **all price files selected in the table as primary data series must contain the same custom indicators** (i.e., the same columns and specifications). Otherwise, the build process will fail when trying to apply the custom indicator(s) to markets for which the required data are not found.

Symbol Settings

As explained in Adding a Symbol, above, when adding a new symbol to the symbol library or viewing the properties of an existing symbol, the Symbol Properties window is displayed, as shown in Fig. 3.4.

This is where all the properties describing the symbol are entered. Click the name of each property to display a description at the bottom of the properties window, as shown in Fig. 3.4 for Cost Type. Most properties can be edited directly by clicking them.

Please Note: When working in the Symbol Library, this window represents the intrinsic properties of the symbol. Once defined, these properties should not need to be changed. When building or evaluating strategies, different values for the bar type, bar size, and trading costs can be specified on the Build Symbols and Evaluation Symbols windows through the entries on the Market Symbols table to suit your purposes at that time. Doing so will have no effect on the properties entered via the Symbol Library. The Symbol Properties window can also be opened through the Build Symbols and Evaluation Symbols windows by double-clicking on the symbol entered in the Market Symbols table. In this case, changes made to the settings on Symbol Properties will not affect those in the Symbol Library.

Symbol
A text label can be entered for the Symbol setting to help identify the market and to provide a label for use in displaying results. Any valid text string can be entered. The symbol is used in the build report, trade list, and for other display purposes. The symbol name is also used when multiple data series have been specified to determine if two symbols in different series are the same.

Description
Enter a text label for the Description setting to help identify the market.

Symbol Type
If the symbol represents an external equity curve, set this property to "Equity Curve". Otherwise, it can be set to "Stocks", "Futures" or "Forex" for price symbols. When this property is set to Equity Curve, the only other properties displayed on this window are Symbol, Description, Data File Name, First Available Date, and Last Available Date, which are the only applicable properties for an external equity curve.

Data File Name
The Data File Name specifies the name of the text file of data. Please see Adding a Symbol, above, for details.

First Available Date, Last Available Date
The first and last available date properties display the date range for data read from the price file. These properties are for informational purposes only; they cannot be edited. If the program is not displaying the full date range in the price file, click the Reload button (see below) to reread the file and reset the available date range. To change the range of dates used for building or evaluating a strategy, use the date controls on the Build Symbols or Evaluation Symbols window.

Bar Type, Bar Size, Range Bar Size, Ticks per Bar
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 setting, which is only shown when the Bar Type is Intraday, should contain the number of minutes for intraday data. Range bars represent trading within a specified point range, such as five points. All bars in a range bar chart have the same range. For range bars, a Range Bar Size setting will be available, which should be set to the high-to-low range of the bars in points. For tick bars, a Ticks per Bar setting is displayed, which is the number of ticks composing each bar; e.g., 100 for 100-tick bars. *The number of ticks per bar is only used in AmiBroker code for secondary data series and can be excluded if desired if not building for the AmiBroker platform.*

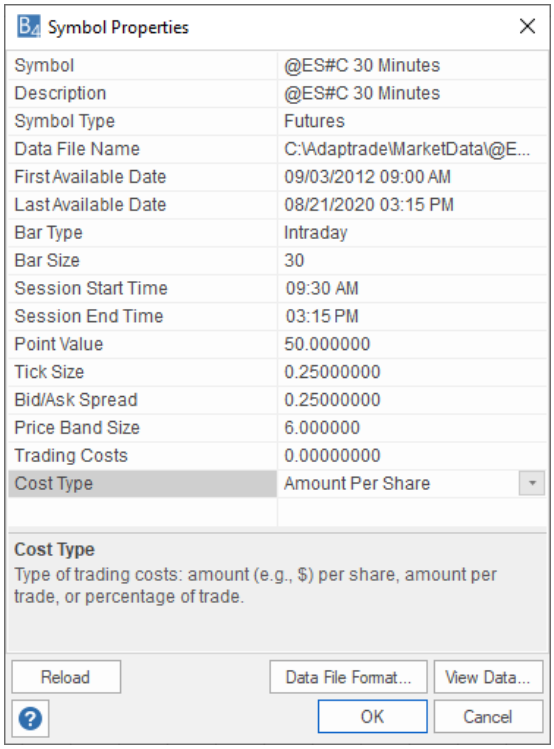


Figure 3.4. Symbol Properties window.

The Bar Type and, where applicable, Bar Size (for intraday data) or Range Bar Size (for range bars) should match the data in the file. 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. For example, if the data in the price file represent 5 min bars, the Bar Type should be "Intraday" and the Bar Size should be 5.

The Daily, weekly, and monthly Bar Type options represent 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. **Single-tick data is not a valid input option in Builder.** Tick bars contain multiple ticks so that each bar has an open, high, low, and closing price.

Session Start Time, Session End Time

The Session Start Time and Session End Time properties specify the trading session times. Only one session is defined in Builder for each symbol, although a combined session, such as day-plus-evening, can be used if the session times encompass both sessions. When accessing the Symbol Properties window through the Symbol Library, these times should exactly match the times found in the price data file. An incorrect session time can affect the end-of-day exit. Builder does not perform time zone conversion.

When accessing the Symbol Properties window through the Market Symbols table of the Build Symbols or Evaluation Symbols window, the session times can be changed without affecting the settings in the Symbol Library. For example, if the price data file contains data for session times spanning 24 hours, as with typical forex symbols, the session times could be changed through the Build Symbols window to daytime hours, such as 8:30 am to 5:30 pm. However, it's important to make sure the data used in the trading platform exactly match the settings made in Builder. In this example, the strategy should be executed on data that use

session times of 8:30 am to 5:30 pm, rather than the original, 24-hour data. Otherwise, much different results may be found in the trading platform than in Builder. To restrict trades to a range of times, it's usually better to use the trade entry and exit time range options under Strategy Logic, rather than changing the session times.

Point Value
Each price symbol should have a valid point value. 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 the value per contract. For stocks and forex, the point value should be set to 1. Note that when forex symbols have a point value of 1, the position size is typically 10,000 or 100,000 for one lot. The point value property has a default value of 1.

Tick Size
The tick size is the smallest allowable price change for the symbol, such as 0.25 for the E-mini S&P 500 or 0.01 (1 cent) for US equities. Builder will infer this from the price data file. If it's incorrect, it can be changed here.

Bid/Ask Spread
The bid/ask spread is the difference between the current price buyers are willing to purchase the instrument for and the price sellers are willing to sell it for. The bid is the lower price, whereas the ask is the higher price. Buy orders are always filled at the ask, and sell orders are always filled at the bid. **A price chart displays the bid price only.** In Builder, the bid/ask spread is only used if the Order Fill Rule option on the Evaluation menu ("Use Bid/Ask spread to determine fills") is checked. If so, market buy orders will be filled above the apparent market price (based on the chart), whereas market sell orders will be filled at the price seen on the chart. The bid/ask spread is part of the cost of the trade.

The bid/ask spread is intended for forex trading and is not typically used for trading futures. When developing forex strategies for MetaTrader 4, this option should be used because MetaTrader 4 uses the bid/ask spread when evaluating strategies. This is consistent with the common practice in forex trading of paying for the trade through the spread rather than paying the broker a fixed commission.

If the bid/ask option is selected, Builder also uses the bid/ask spread to determine if a pending order is filled. For example, a buy stop order is only filled if the ask price, which is above the chart price (bid), touches the stop price. If, for example, the price bar on the chart just touches the stop price, it may appear that the order should be filled, but Builder won't show the historical trade as filled unless the ask price reached the buy stop price. Similarly, a buy limit order will not be recorded as filled unless the ask price reaches down to the buy limit price. Sell stops and limits are filled at the bid, so, unlike buy orders, their fill prices directly correspond to the chart prices. This is also how MetaTrader 4 determines fills.

Note: In addition to the bid/ask spread, MetaTrader 4 applies a minimum price distance to determine if an order can be placed. If a pending order (stop or limit) is too close to the market at the time it's placed, the order will be rejected. This is based on the idea that there won't be sufficient time to place the order before the market moves through the order price. Builder doesn't reject such orders, which can sometimes cause a discrepancy in back-testing between Builder and MetaTrader 4.

Price Band Size
Chicago Mercantile Exchange (CME, www.cmegroup.com) futures use price bands to determine whether limit orders are acceptable. If a limit price is outside the price band, the limit order will be rejected. The size of the price band is given in points, such as 6 points for the E-mini S&P 500 futures. If the price band option under Limit Order Options (Evaluation

menu) is selected ("Place limit order only if order price is within market's price band"), a limit order will only be placed if the order price is within the number of points of the bar's close given by the price band size.

Trading Costs, Cost Type
Trading costs are specified according to the Cost Type. There are three Cost Type options: Amount per Share, Amount per Trade, and Percent of Trade. If the selected cost type is "Amount per Share", the trading costs are calculated by multiplying the entered cost value by the number of shares. For the "Amount per Trade" cost type, the trading costs are simply the entered value. If "Percent of Trade" is selected as the cost type, the trading costs are calculated by multiplying the entered value as a percentage by the trade value, which is the product of the entry price, position size, and point value. In all cases, the calculated trading costs represent the total trading costs (commissions, fees, and slippage) deducted from each trade on a round-turn basis (i.e., deducted once per 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 into Builder.*

Trading costs should always be included though they need not be precise. The purpose is to ensure 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.

This property represents the default value for costs that is shown when adding the symbol to the Build Symbols window for building strategies or the Evaluation Symbols window for evaluating strategies. The cost value can be changed through the Market Symbols table of the Build Symbols and Evaluation Symbols windows.

Reload Button
The Reload button on this window scans the price file for the first and last available dates and updates those fields.

Data File Format Button
The Data File Format button opens the Data File Format window, which enables you to edit or change the format specifications for the data file, as explained in Adding a Symbol.

View Data Button
The View Data button opens a tabular display of the contents of the symbol’s data file. If the symbol represents an external equity curve, the View Data window displays the equity curve changes, rather than the equity curve values. The equity curve values within the equity curve file are unchanged by this process.

Build and Evaluation Symbols

When building trading strategies in Adaptrade Builder, the strategies are applied to the market symbols in the Market Symbols table of the Build Symbols window, as shown in Fig. 3.5. Similarly, when evaluating strategies separately from the build process, the symbols for the back-test (evaluation) are selected through the Evaluation Symbols window, as shown in Fig. 3.6. These two windows are largely the same and work mostly the same except that the training/test/validation settings are not available on the Evaluation Symbols window since those settings only apply when building strategies.

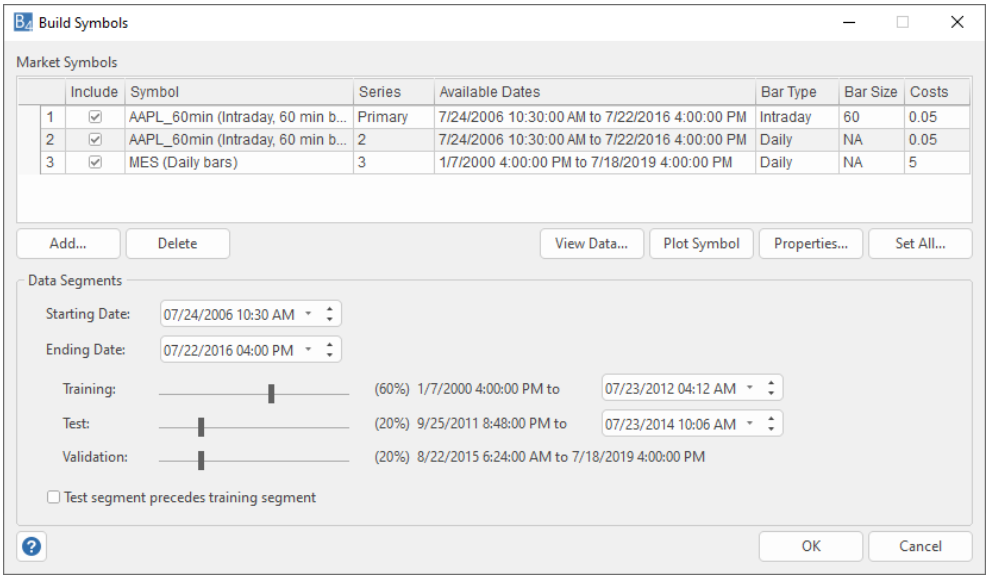


Figure 3.5. The Build Symbols window is used to specify the markets and settings for building strategies.

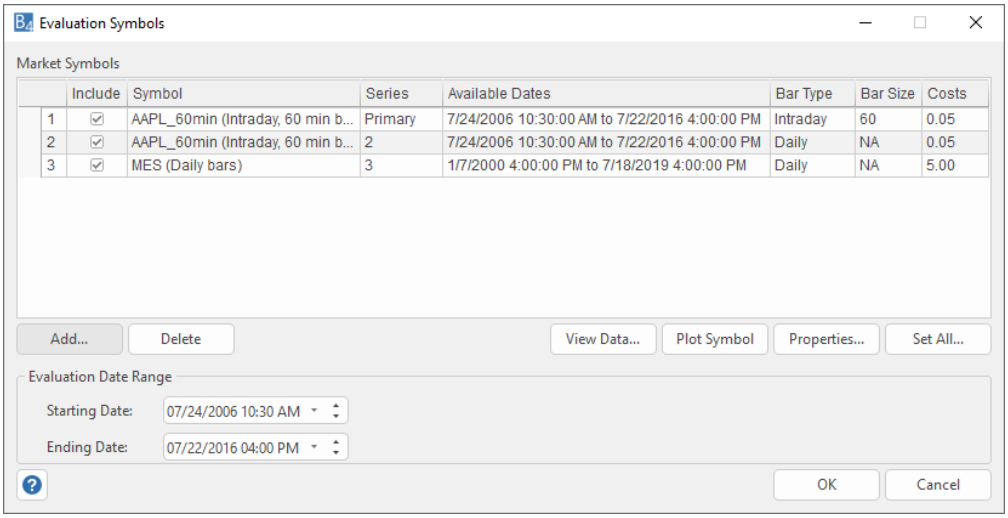


Figure 3.6. The Evaluation Symbols window is used to specify the markets and settings for evaluating strategies separately from building.

Once symbols have been added to the Market Symbols table of the Build Symbols window (see below), the data series for each symbol should be set using the pull-down menu in the Series column. The symbol or symbols on which the strategies will trade are the "Primary" data series. All other series are "secondary" and are numbered from 2 to as high as 20. Symbols are added as primary by default, and any number of primary symbols can be added. When there are multiple primary symbols, the strategies are evaluated on each such symbol, the trades from all markets are combined, and the results are reported as a portfolio.

The secondary symbols provide other data series for the strategy logic. For example, the primary symbol could be the E-mini NASDAQ futures, and a secondary series could be a forex symbol, such as EURUSD. This would enable the program to include indicators applied to EURUSD in addition to ones involving the E-mini NASDAQ.

Secondary symbols can be on different timeframes than the primary market(s), either larger bar sizes or smaller ones.* Also, the same symbol can be selected as both a primary and secondary symbol but with different timeframes. For example, you could select 5-minute bars of AAPL as the primary data series with daily bars of AAPL as data series 2. If the two symbol names are identical (i.e., both are shown as "AAPL" under the Symbol column), then the program may generate logic that compares indicators between the two series, such as "Average(C, N1) < Average(C, N2) of data2". Comparisons between different series are only allowed if the symbol names are the same.

* *The only exception is for AmiBroker code, which generally requires the primary market to be a smaller bar size than any secondary symbol. If this is not the case and AmiBroker has been selected as the code type for building, a warning message will be displayed when the build process is started.*

If there are multiple primary symbols and one or more secondary symbols, the same secondary symbols will be used when evaluating/back-testing the strategy on each primary symbol. For example, you might have a small basket of stock symbols, each of which is a primary symbol. In addition, you could have a stock index and an interest rate as secondary symbols (data series 2 and 3). In that case, when the strategy is back-tested (e.g., during the build process), it will be evaluated on the first stock symbol using the interest rate and stock index symbols, then on the second stock symbol using the interest rate and stock index symbols, and so on for each of the primary symbols.

Note: When executing strategies in the trading platform, the same data for all series must be available in the same configuration as specified in the Build Symbols window. For example, for TradeStation and MultiCharts, secondary data series must be plotted on the chart of the primary symbol in the correct order as data2, data3, etc.

Tip: When the Evaluation Symbols window is first opened, it lists the same symbols as in the Build Symbols window. If the symbols in the Build Symbols window are later changed, the symbols in the Evaluation Symbols window may no longer match those in the Build Symbols window. If you want to easily change the selections in the Evaluation Symbols window to match those in the Build Symbols window, simply open the Evaluation Symbols window, drag the mouse across all symbols in the table to select them, click the Delete button, then close the window and reopen it. This will reset the symbols in the Evaluation Symbols window to those in the Build Symbols window.

As explained below, any symbol in the symbol library can be added to the Market Symbols table via the Add button. Once symbols have been added to the table, you can change the bar type, bar size, and costs per share settings as desired for purposes of building or evaluating strategies. Changes here have no effect on the symbol's properties, which can be seen by double-clicking the symbol or clicking the Properties button to open the Symbol Properties window. Symbols in the Market Symbols table are copies of those in the symbol library, so any change to the settings made here has no effect on the properties in the symbol library.

Note: For equity curve symbols, the Bar Type, Bar Size, and Costs Per Share entries will be shown as “NA” since these properties are not applicable to equity curve symbols.

To change the symbol settings to the values you want to use when building or evaluating a strategy, simply click on the entry in the Market Symbols table. For example, if the data file contained 30-minute bars, you could change the bar type to daily bars for building and/or evaluating strategies on that time frame.

Any valid bar type selection can be made, provided the price data in the file can be converted to the selected type. Price data provided as 1-minute intraday bars allow for the most flexibility because they can be converted to any other intraday size, as well as to range bars, daily, weekly, and monthly bars. No data type can be converted to tick bars since none of the other data types includes the number of ticks in the bar.

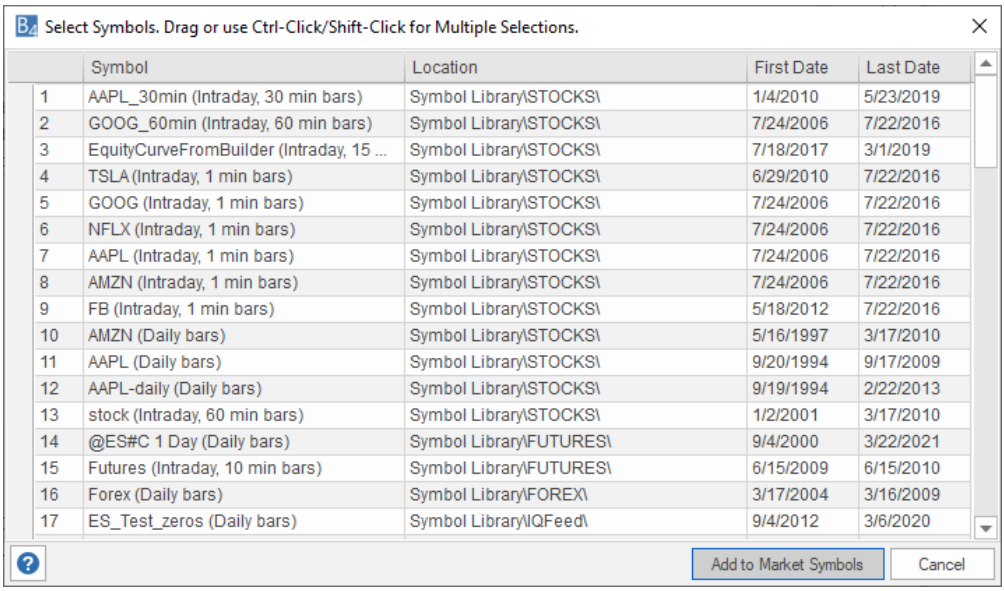
The program will attempt to convert the available data to the selected bar type and will display a warning message in the Messages window if the conversion may not be accurate. For example, if the price data are provided as tick bars and the selection is for 5-minute bars, there is no way to guarantee that accurate 5-minute bars can be constructed because the tick bars may not be smaller than 5-minute bars and/or the tick bars may not close at the precise closing time of the 5-minute bars. As another example, converting from 3-minute bars to 10-minute bars will not be accurate because there is no even multiple of 3-minute bars that can be used to construct a 10-minute bar. *It's the user's responsibility to ensure the selected data settings are sufficiently accurate and that they match the data used in the target platform when executing a strategy in real time.*

Note: When selecting the range bar size for range bars, the size should be large enough compared to the interval of the available data to capture the normal oscillations of the data. For example, if the available price data consist of 5-minute bars, the size of the range bars should be as large as the normal range of movement that occurs within 5 minutes. Otherwise, more range bars will be formed in real time than in back-testing.

Please note that the Symbol field in the Market Symbols table is a property of the symbol, so it cannot be changed directly in the table. If you want to change the symbol label, double-click on the symbol or click the Properties button to open the Symbol Properties window. Similarly, the Available Dates field is informational only; it shows the available dates in the price data file. To change the date range for building or evaluating strategies, use the date/calendar controls.

Add button
To add a symbol to the Market Symbols table, either use the Add to Project command or button in the Symbols pane or click the Add button below the Market Symbols table, as shown in Figs. 3.5 and 3.6. Clicking the Add button opens the Select Symbols window shown in Fig. 3.7. This window displays all available symbols in the symbol library. You can select multiple symbols by dragging the mouse across the symbols or by using the Ctrl-Click or Shift-Click key/mouse combinations. The selected symbols will be added to the Market Symbols table.

When multiple markets are in the Market Symbols table, any or all of the markets in the table can be selected by checking the box in the Include column for the respective market. For a market to be included in the build process, either as a primary or secondary data series, the box in the corresponding Include column must be checked. Any unchecked symbols will be ignored by the build process.



	Symbol	Location	First Date	Last Date
1	AAPL_30min (Intraday, 30 min bars)	Symbol Library\STOCKS\	1/4/2010	5/23/2019
2	GOOG_60min (Intraday, 60 min bars)	Symbol Library\STOCKS\	7/24/2006	7/22/2016
3	EquityCurveFromBuilder (Intraday, 15 ...	Symbol Library\STOCKS\	7/18/2017	3/1/2019
4	TSLA (Intraday, 1 min bars)	Symbol Library\STOCKS\	6/29/2010	7/22/2016
5	GOOG (Intraday, 1 min bars)	Symbol Library\STOCKS\	7/24/2006	7/22/2016
6	NFLX (Intraday, 1 min bars)	Symbol Library\STOCKS\	7/24/2006	7/22/2016
7	AAPL (Intraday, 1 min bars)	Symbol Library\STOCKS\	7/24/2006	7/22/2016
8	AMZN (Intraday, 1 min bars)	Symbol Library\STOCKS\	7/24/2006	7/22/2016
9	FB (Intraday, 1 min bars)	Symbol Library\STOCKS\	5/18/2012	7/22/2016
10	AMZN (Daily bars)	Symbol Library\STOCKS\	5/16/1997	3/17/2010
11	AAPL (Daily bars)	Symbol Library\STOCKS\	9/20/1994	9/17/2009
12	AAPL-daily (Daily bars)	Symbol Library\STOCKS\	9/19/1994	2/22/2013
13	stock (Intraday, 60 min bars)	Symbol Library\STOCKS\	1/2/2001	3/17/2010
14	@ES#C 1 Day (Daily bars)	Symbol Library\FUTURES\	9/4/2000	3/22/2021
15	Futures (Intraday, 10 min bars)	Symbol Library\FUTURES\	6/15/2009	6/15/2010
16	Forex (Daily bars)	Symbol Library\FOREX\	3/17/2004	3/16/2009
17	ES_Test_zeros (Daily bars)	Symbol Library\IQFeed\	9/4/2012	3/6/2020

Figure 3.7. The Select Symbols window allows you to add one or more symbols from the symbol library to the Market Symbols table of the Build Symbols or Evaluation Symbols window.

Delete Button
Symbols can be deleted from the Market Symbols table either by clicking the Delete button or by pressing the Del key.

View Data Button
Click the View Data button to display a window listing the data that have been read. The View Data window displays the data in the format selected in the Market Symbols table. For example, if the data file contains 1-minute bars, but the Bar Type in the Market Symbols table has been changed to Range, range bars will be shown in this window.

If the symbol represents an external equity curve, the View Data window displays the equity curve changes, rather than the equity curve values. This does not change the contents of the equity curve file.

This window also allows you to save the data to a csv (comma-delimited text) file via the File menu or copy the data to the clipboard via the Edit menu. If the Copy command is chosen, the data will be copied in tab-delimited format, which permits it to be pasted directly into a spreadsheet.

Plot Symbol Button
To create a price chart displaying the selected symbol in the Market Symbols table, click the Plot Symbol button.

Properties Button
Click the Properties button to display the Properties window for the currently selected symbol. If no symbol is selected – for example, checked but not selected – then the window will not be displayed.

Set All Button
To set the properties for multiple symbols together -- for example, to set the trading costs to $5 for all selected symbols -- select multiple rows in the table and click the Set All button.

Date Range Settings

The Build Symbols and Evaluation Symbols windows also include controls for changing the start and end dates and times for analysis. To change the start and end dates for building or evaluating a strategy, click on the calendar controls or select the part of the date or time you want to change and type in the new value. To change the year, enter the last two digits only.

The Build Symbols window includes controls for changing the percentage of data used for training, test, and validation. Move the sliders to change the percentage of the date range allocated to each segment. The training data are used in building the strategy. Test data are used in evaluating the build process while it's ongoing. Data in the validation segment are used for evaluating the strategies after building. The validation segment is "out-of-sample", which means the data is not used in the build process, and the results on this segment therefore provide an objective test of the strategies.

As an alternative to the slider controls and to provide more precision in setting the segment boundaries, there are date/time controls for the end of the training and test segments. Changing the date and time for either segment will redefine the end of the segment, inclusive of the chosen values. The slider positions and percentages will be updated after a change is made to the control. The validation segment does not have a date/time control because it's defined as the remainder of the total period following the training and test segments.

The check box below the slider controls allows you to specify whether the test period follows the training period or precedes it. Leave the box unchecked if you want the training period to come first or check the box if you want the training period to follow the test period. The validation period is always the most recent segment of data.

Builder stores the dates over which the strategies were built (i.e., training, test, and validation) in the project file (.gpstrat). Once the strategies in a project file have been built, the build dates cannot be changed unless you rebuild the strategies. If you change the date range on the Evaluation Symbols window after building and then evaluate a strategy (i.e., by selecting Evaluate from the Evaluation menu), Builder will label the different segments of the equity curve as "OOS Pre", "In-Sample", and "OOS Post". OOS Pre corresponds to the portion of the selected date range, if any, that precedes the date range used in building (training, test, and validation). In-Sample corresponds to the date range used in building, and OOS Post corresponds to any data following the date range used in building.

If multiple markets have been selected in the Market Symbols table, the date range is restricted to the range of the primary markets since those are the markets on which trading takes place. The default dates shown in the calendar controls represent the earliest starting date and latest ending date over all selected primary markets.

Chapter 4

Price Charts

Creating and Using Price Charts

Project files in Adaptrade Builder can contain one or more price charts, which are displayed in tabbed windows, as shown in the figures below. Price charts display the price data as open/high/low/close bars with volume as a separate bar graph. When building a population of deep neural networks (DNNs) or when one is included in the currently selected strategy, the DNN prediction is superimposed as a line over the price bars, as shown below in Fig. 4.1. A data box is shown when the cursor is placed over a price bar, as shown in the figure. The data box displays the price data and, if a DNN is present, the DNN value for the current close and the DNN's prediction made at that close. For example, if the DNN has been trained to predict the price 10 bars ahead, the "DNN Predict" value will be for the close 10 bars ahead.

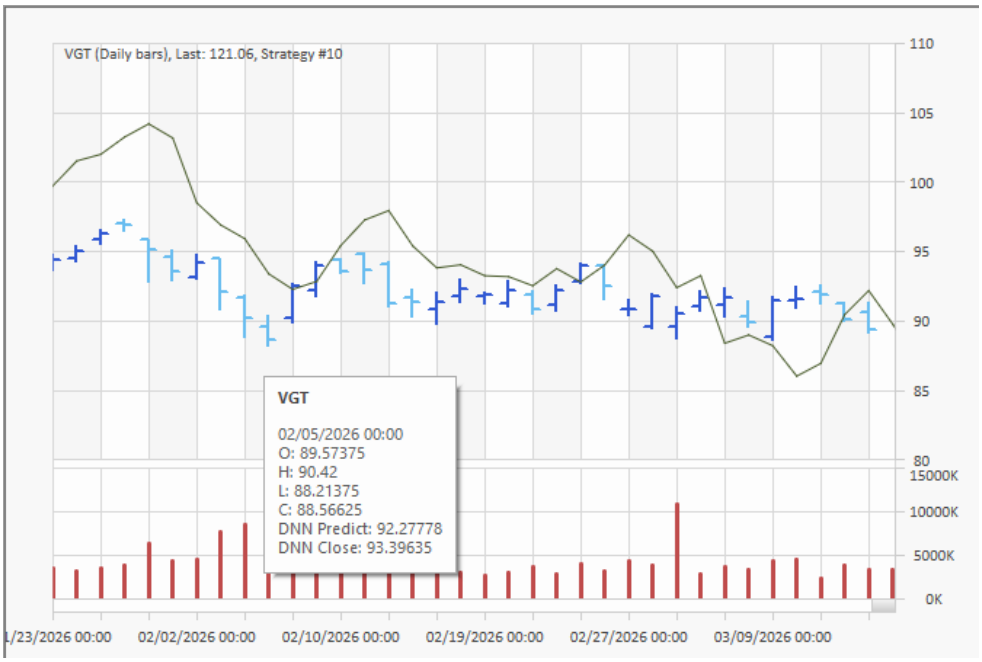


Figure 4.1. A price chart with a DNN prediction line. The data box displays the price data and the DNN results at the selected bar.

The price chart can also display the trades from the currently selected strategy, as shown below in Fig. 4.2. To view the trades, select the **Show Trades** command from the context (right-click) menu, which will display the trade entry and exit markers on the price chart provided the strategy currently selected in the Build Results table contains trades for the same symbol.

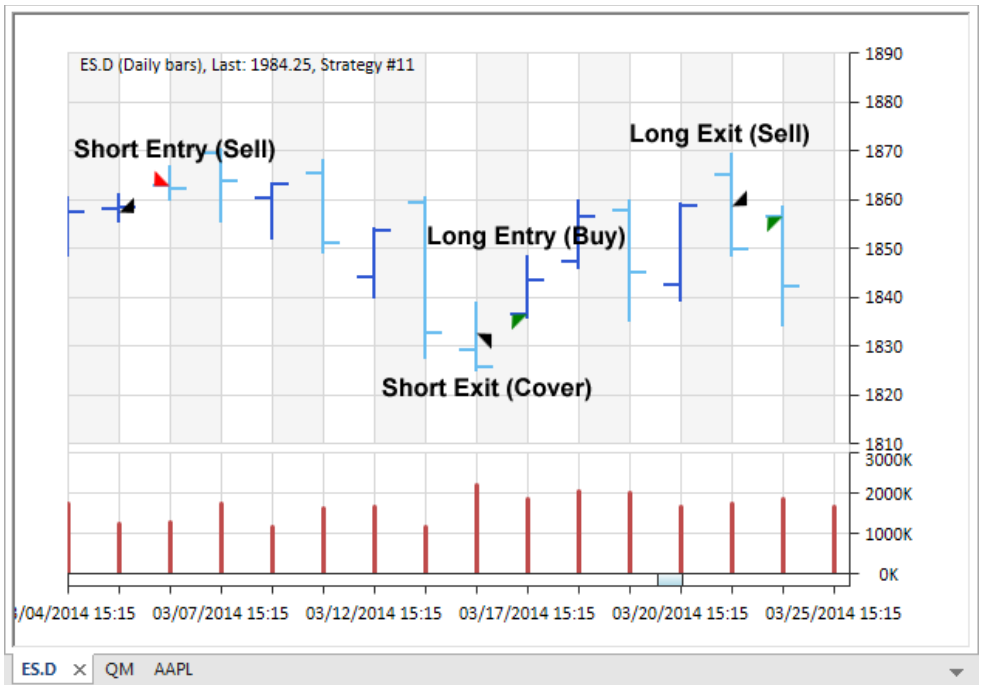


Figure 4.2. Trade entry and exit markers on a price chart.

The triangular trade markers can be distinguished by both color and shape. There are four different markers: short entry, short exit, long entry, and long exit. The orientation of the triangle is meant to signify the trade direction and type of order (entry or exit). Entry markers are on the left side of the price bar, whereas exit markers are on the right side of the bar. The long entry and exit markers point upward, whereas the short entry and exit markers point downward. Long entry markers are green, short entry markers are red, and both exit markers are black.

If a different strategy is selected in the Build Results tables, the price chart will be updated automatically to display the trades for the selected strategy, provided the symbols still match. This makes it easy to review the trade entries and exits for different strategies on the price chart. If the selected strategy has been applied to multiple symbols, only the trades for the symbol matching the chart are displayed. Charts for the other symbols can be opened to show the trades for those symbols.

The other commands on the context menu for charts are: **Show Ideal Trades**, **Ideal Trade Settings**, **Show Orders**, **Reload**, and **Select Price File**.

The **Show Ideal Trades** command displays the so-called ideal trades on the price chart. Ideal trades represent hypothetical ideal entry and exit locations for trades and are used in combination with the Ideal Matches metric. Basically, the Ideal Matches metric represents the correspondence between the strategy trades and the ideal trades as a percentage, with 100% representing perfect correspondence. Ideal trades and the Ideal Matches metric are described in detail in the Usage Topics chapter, Ideal Trades.

Ideal trades are shown below in Fig. 4.3. The green horizontal bars with up-turned ends positioned below price bars represent the entry locations of long ideal trades, and the red bars with down-turned ends positioned above price bars represent the entry locations of short ideal trades. The black dotted lines represent the exit locations; long exits are positioned above the price bars and are drawn with down-turned ends, and short exits are positioned

below the price bars with up-turned ends. Black dotted exit lines drawn without a right-end mark indicate that the exit bars for the ideal trade continue past the end of the line. As explained in the Usage Topics chapter, entry and exit locations for ideal trades can last for several bars, and the exit locations can extend for much longer, as depicted in Fig. 4.3, provided the minimum profit requirement has been met on at least one bar and the maximum risk requirement has not been exceeded.

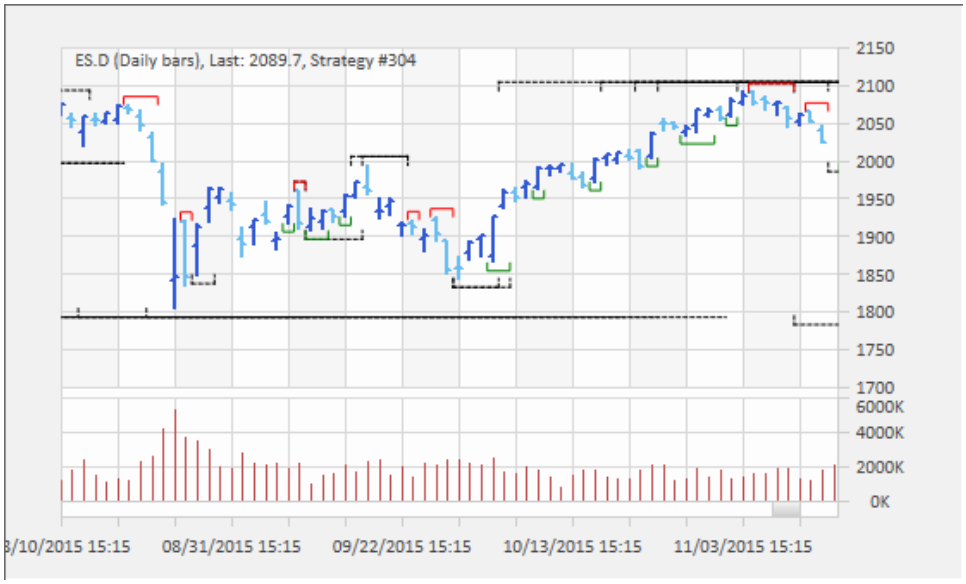


Figure 4.3. Ideal trade markers on a price chart.

To change how the ideal trades are defined, select the context menu command **Ideal Trade Settings**. This command opens the Metric Options window, as shown below in Fig. 4.4. The "Market sides" pull-down menu allows you to select whether the ideal trades are for long trades only, short trades only, or both long and short trades. There are two ways to define the ideal trades: based on fixed size trade amounts ("Fixed size ideal trades") or based on multiples of the average true range ("Ideal trades adaptive to ATR"). In either case, the trades are defined by a minimum profit and a maximum risk or MAE (maximum adverse excursion). For the fixed size option, enter the dollar (or account currency) amounts for the profit and risk on a per-share or per-contract basis. For the adaptive (ATR-based) option, enter the multiples of ATR for the profit and risk and the averaging length for the ATR in bars (e.g., 14 bars). In this case, the profit and risk values will be calculated as the current value of ATR multiplied by the respective entered multiple times the point value for the symbol. When changes to the settings on this window are made, the ideal trades displayed on the chart, if any, will be updated automatically.

Note: The number of ideal trades in each segment of the price data, along with the ideal trade settings in effect at the time the population was built, are displayed in the Build Report.

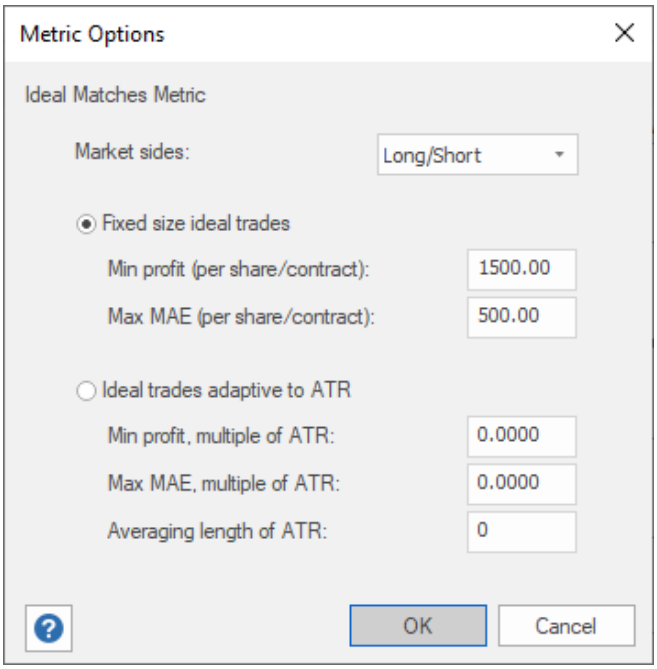


Figure 4.4. Metric Options window for changing the ideal trade settings.

The **Show Orders** command displays the Trading Orders window, which displays the trading orders for the currently selected strategy. The Trading Orders window is discussed in the next section.

The **Reload** command causes the program to read the file of price data and update the chart display. Also, if the option to show the strategy trades has been selected, the selected strategy is re-evaluated and the trade entry and exit markers are updated. If the option to show the ideal trades has been selected, these are updated as well. This command can be useful when using Builder as a trading platform. When new price data come in, the reload command can be used to update the chart along with the orders in the Trading Orders window.

Provided the end date in the Date Range section of the Evaluation Symbols window is set to the last available date shown in the Market Symbols table, the date range for analysis will be automatically updated to the new last date in the price data file, and the strategy evaluation will include this new data. If the Trading Orders window is open, the trading orders will be updated as well. If, for some reason, the last date for analysis is not set to the last available date, the date controls should be used to set the date to the last available date prior to selecting the Reload command. Otherwise, the trading orders will not be updated.

The **Select Price File** command allows you to select a new file of price data for the chart. This can be useful if the file of price data has been moved or its name has been changed.

There are two ways to add a price chart to the project file. From the Symbols pane, select the price symbol in the symbol library and click the Plot Symbol button. Alternatively, select a symbol from the Market Symbols table on the Build Symbols or Evaluation Symbols window and click the Plot Symbol button on that window. In either case, the Plot Symbol command is also available in the context menu. To remove a chart from the project file, click the close button on the chart tab.

When a price chart is created, it includes all the data read from the associated price file. The chart can be scrolled left or right by holding down the left mouse button and dragging left or

right. Clicking the scroll bar at the bottom of the chart will move the chart to that position. The bar spacing can be increased or decreased by moving the mouse scroll wheel. Other formatting options and features may be added in later versions of Builder.

Trading Orders

The Trading Orders window displays the trading orders for the currently selected strategy, as shown in Fig. 4.5.

The trading orders are for the strategy currently selected in the Build Results tables as of the last date of analysis. This is the last date in the date range selected in the Date Range section of the Evaluation Symbols window. The strategy is identified by its member number, shown in the title bar of the window (e.g., #36 in Fig. 4.5). If the strategy was last evaluated on multiple symbols, orders for all symbols will be included in the window.

The Trading Orders window contains two sections: Open Positions and New Orders. Any open positions are shown in the first section. If there is no open position for a symbol, it will be shown as "Flat". Open positions are either "Long" or "Short". The "from" price is the entry price for the position.

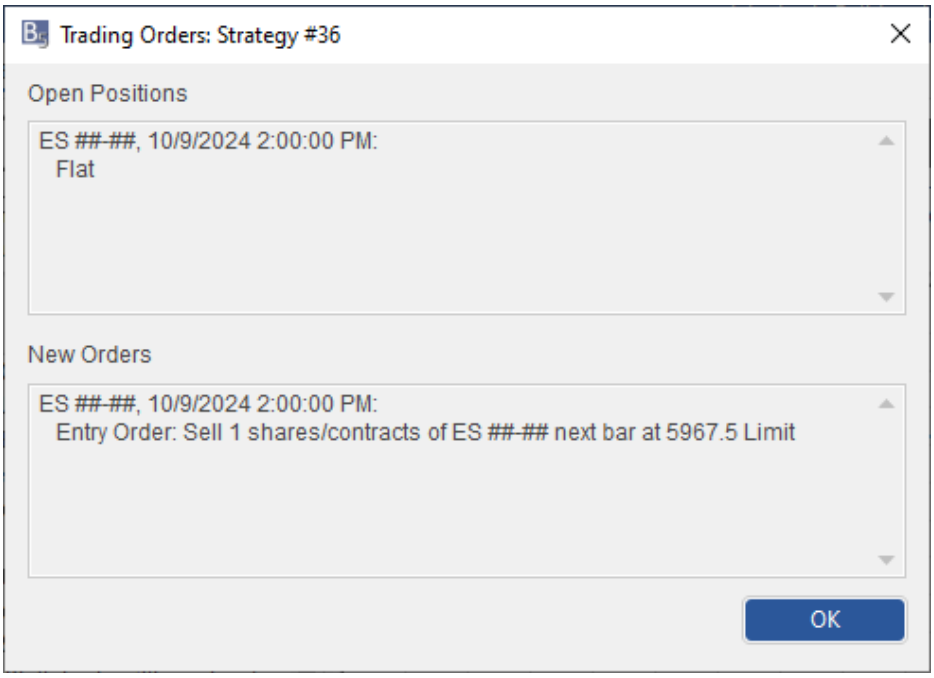


Figure 4.5. The Trading Orders window displays the trading orders for the selected trading strategy.

The New Orders section contains any trading orders that should be placed for the next bar. The orders are shown as either "Exit Order" or "Entry Order". The date and time shown for all listed positions and orders is the date of the last bar of data analyzed.

The trading orders can be selected and copied using the context (right-click) menu by clicking "Select All" then "Copy" or using the Ctrl+C command to copy the orders to the clipboard. After copying, the new orders can be pasted into an email message to your broker.

The Trading Orders window can be opened or closed by selecting "Trading Orders" on the View menu in the ribbon. Alternatively, it can be displayed by right-clicking on a price chart and selecting "Show Orders". The window can be kept open while working elsewhere in the

program. As explained in Creating and Using Price Charts, above, the trading orders can be updated when the data in the price file changes by selecting the Reload command of the price chart context (right-click) menu. The orders will also be updated if the selected strategy is re-evaluated by selecting Evaluate from the Evaluation menu.

Position Sizing

The trading orders include the position size of open positions and the position size for new orders. The position size is based on the position sizing settings currently in effect. For open trades, this will be the same size as shown in the Trade List table for the last trade in the list. Builder has two options for calculating the equity used for position sizing: accumulating the equity trade-to-trade during the back-test or using the equity value set on the Evaluation menu as the "Starting Equity" value; see Position Sizing Settings in the Build Settings chapter. If the option to accumulate the equity is chosen, the size for the most recent trade will be based on the accumulated equity over the entire back-test period, which may be a much higher value than the equity in your trading account. For this reason, to get the correct position size for your current account equity, select the option "Use entered equity value for each trade" on the Position Sizing menu and set the Starting Equity value to the current value of your account equity.

Please Note: Reversal entries don't include the exit order for the open position when listing the orders under New Orders. A reversal entry, such as found in a stop-and-reverse strategy, is one in which there is no separate exit, so that the entry reverses the current position. For example, in a stop-and-reverse strategy, all the entries are reversal entries, and the trades alternate between long and short positions. You are always in the market in this case. With a reversal entry, only the entry order for the next trade is shown in New Orders. The exit order for the current position is not shown. As a result, the entry order will list the position size for the new entry only. In actuality, to reverse an open position, you would need to place an order that includes twice as many contracts or shares. For example, if you're long 3 contracts and the new order is to sell short 3 contracts, you would need to sell short 6 contracts: 3 to close out the long position and 3 to enter the short trade. In this case, the Trading Orders window would only show a new order to sell short 3 contracts, and you would need to recognize that it's a reversal entry and adjust the position size accordingly.

Chapter 5
Build Settings

Before starting the build process, a variety of settings can be made to guide the process and influence the types of strategies or deep neural networks (DNNs) generated during building. Most of these settings and options are selected on the Build menu of the ribbon, shown previously in Fig. 2.9 in Quick Start Steps. The remaining settings are made on the Evaluation (Fig. 2.10) and Position Sizing (Fig. 2.11) menus.

When building strategies, the settings can be made by starting at the left side of the Build menu with the GP Settings button and working left to right, clicking each button in turn. For DNNs, most of the settings are made on the DNN Settings window.

Most of the buttons have an associated window that drops down when the button is clicked, such as the GP Settings window (Fig. 2.5). On the right is the Market Sides setting, which allows you to restrict strategy entries to either long-only, short-only, or both long and short trades. The settings and options for the other buttons of the Build menu and for the Evaluation and Position Sizing menus are described below.

When all the settings have been made, click the **Build** button in the Build Strategies panel of the Build menu to start building trading strategies. To start building a population of DNNs, click the **Build** button in the Market Prediction panel of the Build menu. For either strategies or DNNs, the **Pause/Resume** button in the associated panel can be used to pause the build process temporarily without cancelling it. Click the button a second time to resume. The **Cancel** button in either the Build Strategies or Market Prediction panel will stop the associated build process and display the results for the last completed generation.

GP (Genetic Programming) Settings

The options that influence the build process are selected on the GP Settings window, shown in Fig. 2.5 in Quick Start Steps.

Population Settings

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 getting a good result. 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. Smaller population sizes are recommended when building deep neural networks (DNNs) due to the computational complexity and long build times that are common when building and training DNNs.

Subpopulations can also be specified. This feature allows the main population to be subdivided into smaller groups of strategies that are evolved separately. The purpose of subpopulations is to enforce greater diversity by preventing fitter members from dominating the population, which can lead to uniformity among strategies after some number of generations. The number of subpopulations is entered in **Subpopulations** or by clicking the up/down arrow buttons.

Entering "1" as the number of subpopulations is the same as not using subpopulations; i.e., the main population is the only one. If more than one subpopulation is used, the subpopulations are mixed periodically, as determined by the entry in **Mix subpopulations every [] generations**. This means that the strategies are randomly assigned to the different subpopulations after the entered number of generations so that the strategies from different subpopulations can be combined with each other via crossover. To mix or redefine the subpopulations after each generation, enter "1" here. To not mix them at all, enter the maximum number of generations used in the build.

The number of strategies saved in the project file is specified using the option **After building, save [] fittest strategies**. 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.

Note: The number of stored strategies doesn't affect the build process, which takes place over the number of members in the population, regardless of how many strategies you elect to save. However, once the build process is complete, only the saved strategies are available.

The option to **Save generation with the highest average fitness on the test segment** is designed to reduce the risk of over-fitting. When the results on the test segment start to decline, it can indicate over-fitting, which means the strategies may be starting to fit the market noise, rather than the tradable features (i.e., signal). By saving the strategy with the highest fitness on the test segment, this risk may be lessened. When this option is checked, the stored strategies are from the generation with the highest fitness on the test segment, regardless of how many generations were completed. Otherwise, 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 **Reset on Build** 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 equivalent to resuming the build after pausing it.

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, the build includes the best results from the prior build in the new build 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 from scratch.

Advanced GP Settings

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.

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. This applies primarily to point mutation, which operates on an element-by-element basis. During point 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 and exit rules in Builder are represented by tree structures, as explained in Chapter 1; see Fig. 1.3. The **tree depth** is the number of levels in the tree. For example, the tree in Fig. 1.3 has a depth of five. The default value of tree depth in Builder is 3. To increase the potential complexity of the rules, a higher tree depth can be used. Because the tree depth tends to increase over successive generations as a result of the crossover and mutation operators, the subtree pruning mutation operator is used to reduce the size of the trees if the tree depth exceeds this setting. Another way to prevent the entry and exit rules from becoming overly complex is to include the complexity metric as part of the build metrics; see Appendix: Performance Metrics for a description of the complexity metric.

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 and, optionally, a neural network indicator or a DNN. 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.

Build Failure Rules

If the build process is not going well, the build failure rules can be used to reset it and start over. There are two optional rules. If selected, the rules are applied once after the specified generation. For example, you might elect to apply the rules following generation number 10. If either of the rules applies, the build process will be reset and will start over.

The Fitness Failure Rule calculates the average population fitness on the test segment and takes a N-period moving average, where N is the specified number of generations. It compares the moving average for the current generation to the value M generations ago. If the current value is less than it was M generation ago, the build process is reset. N and M can be any number of generations less than the maximum specified under Build Termination Options.

The Performance Failure Rule calculates the average value of the specified performance metric on the test segment over the top N strategies in the population. If the calculated value is less than (greater than) the specified value, the build process is reset.

Note that both rules are based on the test segment performance. This is because the test segment is designed to monitor the training segment. For a good quality build, the results on the test segment should be increasing, just as they normally do on the training segment. These rules can detect when that is not the case, which may imply that the build process should be reset. Also note that because DNNs don't use the strategy build metrics, only the first build failure rule applies to DNNs.

Build Termination Options
The build termination options define the normal conditions under which the build process should be stopped. There are two ways to stop the build process. First, you can specify both the **Minimum Generations** and the **Maximum Generations**. The build process will run for at least the minimum number of generations and for no more than the maximum number, regardless of the other rules.

In general, 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 as many new population members (strategies) as the size of the population. A higher number of generations generally results in better strategies since it gives the build process more time to evolve the population.

Hint: To explore the variety of strategy logic that can be generated by Builder, try setting the maximum 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 second way to stop the build process is to select one or both of the two optional build termination rules. Both rules are designed to detect when the results are no longer improving so that further building would be fruitless at best and might result in over-fit strategies at worst. The first rule stops the build when it detects a decline in the moving average of the population average fitness on the test segment. Specifically, if the moving average is less than it was N generations ago, the build process is stopped. The second rule stops the build when the fitness (i.e., average population fitness) on the test segment falls below its moving average.

Both the Build Failure Rules and the Build Termination Rules are depicted graphically on the Build Progress window while the build process is ongoing. Please refer to Build Progress Window in the chapter Build Results for more information.

Build Metrics

The Build Metrics tab is shown in Fig. 2.6 in Quick Start Steps. This is where you select the build objectives and conditions that will guide the build process for trading strategies. The metrics used for building populations of DNNs are selected on the Deep Neural Network Settings window. Strategies are ranked during the genetic programming process according to the *fitness*, which is a combination of the build objectives and conditions selected on this tab. The objectives are metrics that you want to minimize (e.g., drawdown) or maximize (e.g., net profit). The conditions are expressed as inequality or equality statements, such as a correlation coefficient greater than or equal to 0.95 or complexity less than or equal to 10. The third list box is for selecting conditions for filtering the strategies for the Top Strategies results.

The buttons to the right of each list box present the basic options of adding, editing, and deleting entries from the adjacent list box. These options are also available via a context (right-click) menu. In addition, cut, copy, and paste command are also available, both via the small buttons below the Delete button next to each list and as standard control-key commands (i.e., Ctrl-X for cut, Crtr-C for copy, and Ctrl-V for paste). Entries in the Build Conditions list can be copied and pasted into the Top Strategies list and vice-versa. To select multiple list items use the shift-click and Ctrl-click key combinations. Although it does not affect the fitness calculations, the order of list items can be changed by cutting an item and pasting it back into a different location in the same list.

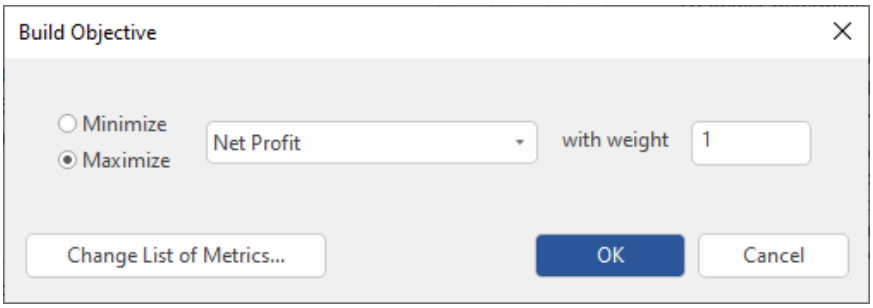


Figure 5.1. The Add and Edit buttons adjacent to the Build Objectives list open the Build Objective window for adding a new build objective or editing the current selection.

To add a new objective to the Build Objectives list, click the Add button adjacent to the list. This will open the window shown above in Fig. 5.1. Select the metric to minimize or maximize and enter the weight value. The weight values are all relative to one another. To weight the metrics evenly, a weight value of 1.0 can be entered for each build objective. The Minimize/Maximize selection will automatically change to the typical choice when the metric is selected. For example, it will change to "Maximize" if you select Net Profit as the metric or to "Minimize" if you select Drawdown as the metric. Click OK to add the new objective or click Cancel to discard it.

To edit a build objective, double-click the metric in the list or click it once and click the Edit button. This will bring up the same window as shown in Fig. 5.1. Make any desired changes and click OK to update the objective in the list or Cancel to leave the objective unchanged.

To add a new condition to the Build Conditions list, click the Add button adjacent to the list. This will open the window shown below in Fig. 5.2. Select the metric for the condition from the pull-down menu, then select the type of relationship (>=, <=, =, or "between"). Finally, enter the value or values. For the "Between" condition, the values can be entered in either order. Click OK to add the new condition to the list or Cancel to discard it. To edit a build condition, double-click the metric in the list or click it once and click the Edit button. This will bring up the same window as shown in Fig. 5.2. Make any desired changes and click OK to update the condition in the list or Cancel to leave the condition unchanged.

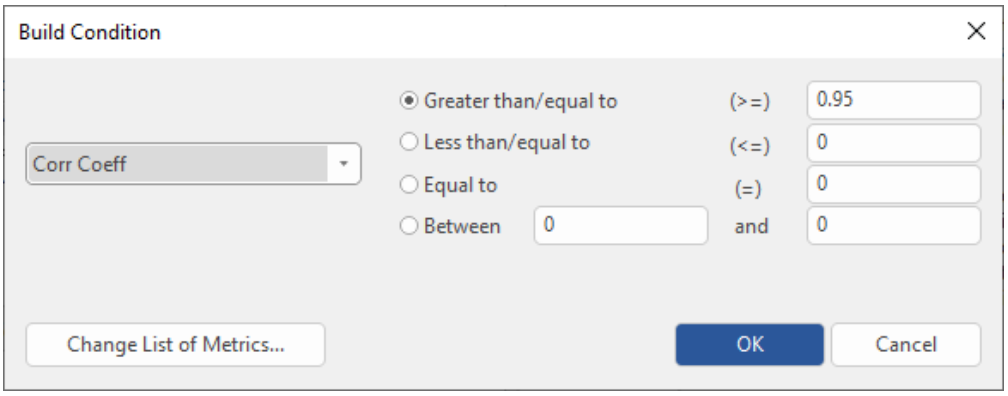


Figure 5.2. The Add and Edit buttons adjacent to the Build Conditions list open the Build Condition window for adding a new build condition or editing the current selection.

The strategy fitness is calculated from a combination of the build objectives and build conditions. The part of the strategy fitness calculated from the build objectives is a weighted sum of the metrics shown in the Build Objectives list. The metrics are normalized to the interval [0, 1] over all members of the population and weighted according to the entered weight values. The resulting values are then scaled so that the maximum value over the population has the value 1. The scaling is performed relative to the initial population, and the same scaling factors are used for each generation so that the fitness values can be compared across generations.

The build conditions are used to *penalize* the fitness when the conditions are not met. The size of the penalty is proportional to the extent to which the condition has not been met. For example, if the condition is that the drawdown should be less than 5000, and the actual drawdown is 7800, the penalty will be 2800, which is then scaled so that the maximum penalty value for that condition over the population members has the value 1. If the condition has been met, the penalty value is zero. Penalty values are subtracted from the fitness. This means that the maximum penalty value that can be subtracted from the fitness is +1 for each build condition. Since the maximum value of the contribution to the fitness from the build objectives is 1.0, the maximum possible value of the fitness is 1.0, which will happen if all of the build conditions are met for the strategy with the maximum value of the build objectives. Negative values of fitness imply one or more build conditions have not been met.

Adding build conditions helps the algorithm converge to a solution in which the build conditions are met. However, it does not guarantee that the build conditions will be met; it's always possible, for example, to specify conditions that exceed the maximum performance potential of the market. It's usually better to select build conditions that represent realistic performance goals. If the metric values representing realistic performance goals are not apparent, a few trial runs can be used to better gauge the potential of the market.

The fitness is calculated over all segments, but only the fitness on the training segment is used by the genetic programming algorithm to evolve the strategies. The fitness value on the test segment is used to monitor the build process according to the rules defined on the GP Settings window.

The values of the metrics used in the build objectives and conditions are the same as the ones listed in the Build Results table. By monitoring the results in the results table, as well as the equity curve, you can get a good idea of which objectives or conditions may need to be added or changed to improve the results. It’s usually best to start with a small number of objectives and conditions, such as three or four. The default selections are often a good

starting point. After a few generations, you can add or remove items, change the weights, and so on based on the reported results. An incremental approach usually works better than selecting objectives and conditions arbitrarily and letting the build process run for an extended period without intervention.

The list labeled **Conditions for Selecting Top Strategies** contains conditions used to filter or select members of the population for the Top Strategies results. If all the conditions in the list are met, the strategy is added to the Top Strategies table. Each member of the population is checked against this list as each generation is evolved. The conditions are defined the same as in the Build Conditions table except that you can specify the data segment over which the condition will be evaluated.

To add a new condition to the Top Strategies conditions list, click the Add button adjacent to the list. This will open the window shown below in Fig. 5.3. Select the metric for the condition from the pull-down menu, then select the type of relationship (>=, <=, =, or "between") and enter the value or values. Finally, select the data segment over which the condition will be evaluated (training, test, validation, or all segments). Click OK to add the new condition to the list or Cancel to discard it. To edit a condition, double-click the metric in the list or click it once and click the Edit button. This will bring up the same window as shown in Fig. 5.3. Make any desired changes and click OK to update the condition in the list or Cancel to leave the condition unchanged.

Please note: The segment "All segments" is the segment consisting of trades from all segments combined into one. The metric values on this combined segment are the ones displayed in the performance report. "All segments" does not mean that the condition will apply to all segments individually. For example, if a condition for the profit factor is true over "all segments", it may or may not be true for the training, test, or validation segments separately. If you want a metric condition to be applied to all segments separately, add the same condition for each segment.

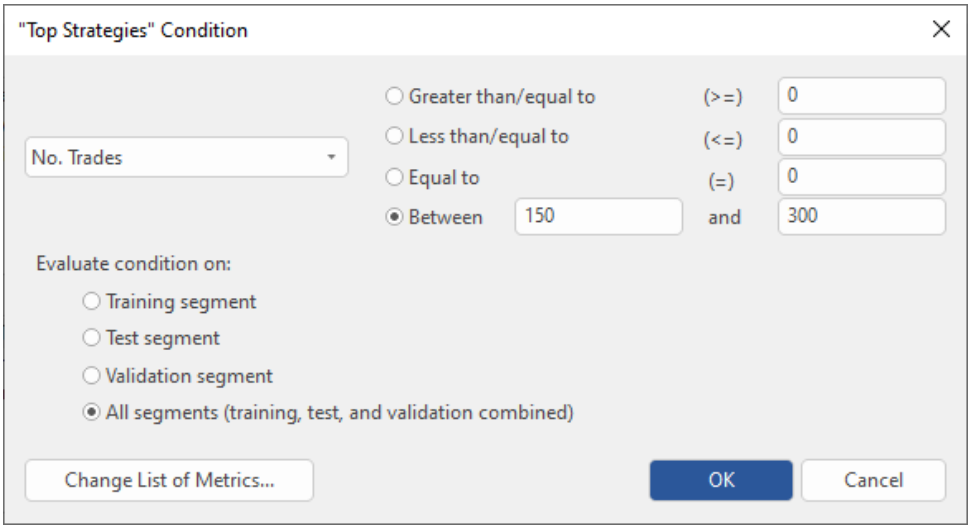


Figure 5.3. The Add and Edit buttons adjacent to the Top Strategies conditions list open the "Top Strategies" Condition window for adding a new condition or editing the current selection.

See Appendix: Performance Metrics for a description of each available metric.

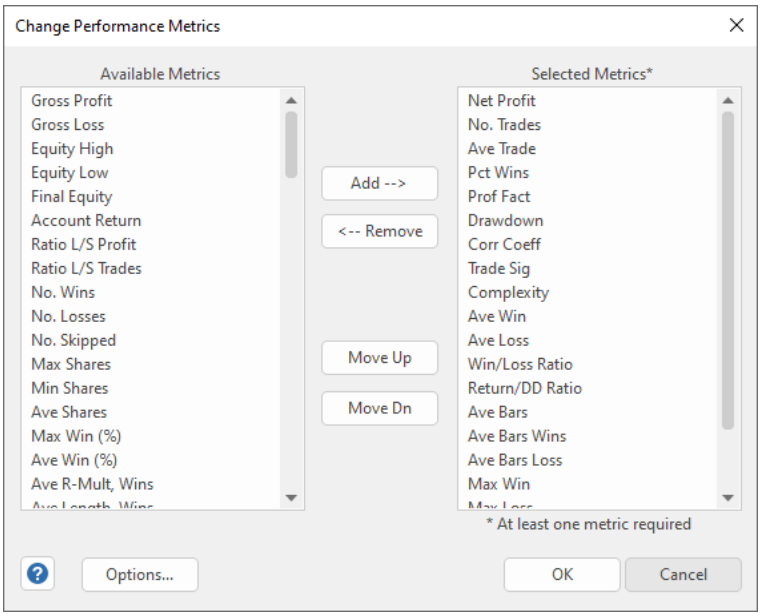


Figure 5.4. The Change Performance Metrics window is used to add or remove performance metrics available on the Metrics tab and shown in Build Results.

The list of metrics available for any of the lists on the Metrics tab can be changed by clicking the "Change List of Metrics" button, which is available on each of the Edit/Add windows (Figs. 5.1, 5.2, 5.3). This will bring up the Change Performance Metrics window, as shown in Fig. 5.4.

To add a metric, select it from the list of "Available Metrics" then click the Add button. The metric will be added to the end of the list by default. If you want to add the new metric to a specific location in the list of selected metrics, select a metric from the right-hand list before adding the new metric. When you click the add button, the new metric will be placed above the metric you selected in the right-hand list. To remove a metric from the list of selected metrics, select it from the right-hand list, then click the Remove button. To move a metric up or down in the list of Selected Metrics, use the Move Up/Dn buttons. The Add/Remove/Move Up/Move Dn buttons can also be accessed by right-clicking. To select multiple list items, use the shift-click and Ctrl-click key combinations or hold down the mouse button while dragging the cursor over the items you want to select.

Changing the metrics here also changes them in the Build Results table and in the pull-down menu for the Performance Failure Rule on the GP Settings window.

The Options button in the lower left-hand corner of the Change Performance Metrics window (Fig. 5.4) opens the Metric Options window, which allows you to select or change the settings for the Ideal Matches metric. Please refer to the Price Charts chapter, Creating and Using Price Charts, for more information on changing the settings for the Ideal Matches metric and to Using Price Charts and the Usage Topics chapter, Ideal Trades, for more information on ideal trades.

Stress Testing

The options for stress testing and Monte Carlo analysis are available on the Stress Testing/Monte Carlo Analysis window, which drops down when the Stress Testing button on the Build menu is clicked, as shown in Fig. 5.5.

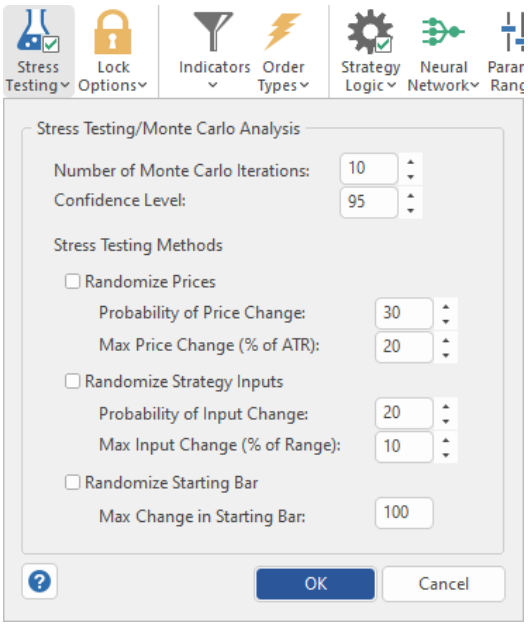


Figure 5.5. Stress testing/Monte Carlo analysis options.

This group of options allows you to select the settings for stress testing and Monte Carlo analysis, which are explained in detail in the chapter Usage Topics. Stress testing involves making multiple sets of small, random changes to the trading strategy and/or market prices and evaluating the results using Monte Carlo analysis.

Stress testing can either be applied during the build process or after-the-fact when re-evaluating a strategy using the Evaluate command. To apply stress testing as part of the build process, make sure the option "Apply stress tests/Monte Carlo analysis" is selected on the Evaluation menu and that at least one of the stress testing methods is selected.

The number of Monte Carlo iterations is the number of evaluations made for each Monte Carlo analysis in addition to the analysis for the original data and settings. Each iteration is a separate stress test, in which the parameters of the test are randomly chosen according to the settings under Stress Test Methods. The equity curve corresponding to each stress test will be displayed in the Equity Curve window; for example, if 10 is entered for the number of Monte Carlo iterations, 11 equity curves will be shown in the Equity Curve window, one for each stress test plus the original equity curve. The results of all iterations are reported at the confidence level entered for Confidence Level. For example, if a value of 95 is entered, the Monte Carlo results of the stress tests will be reported at a confidence level of 95%.

There are three optional stress testing methods: randomizing prices, randomizing strategy inputs, and randomizing the starting bar. Check the box next to each method to include it as part of the stress test for each Monte Carlo iteration. At least one box must be checked for a valid stress test.

To randomize the prices, check the box and enter values for the probability of changing a price and for the maximum amount by which the price is changed. The probability is applied to determine whether to change each price (open, high, low, close). If a price is changed, it will be changed by a randomly chosen amount not greater than the maximum amount entered as a percentage (positive or negative) of the average true range.

To randomize the strategy inputs, check the box and enter values for the probability of changing the inputs and for the maximum amount by which an input is changed. The probability is applied to determine whether to change each input. If an input is changed, it will be changed by a randomly chosen amount not greater than the maximum amount entered as a percentage (positive or negative) of the range for that input. The range for each input is determined from the ranges set on the Parameter Ranges drop-down window (Build menu). When applied to building a population of DNNs, this option refers to the parameters of the DNN's inputs.

To randomize the starting bar, check the box and enter a value for the maximum change in the starting bar. If the starting bar is changed, it will be changed by a randomly chosen value not greater than the value entered.

Lock Options

The Lock Options window, which drops down when the Lock Options button on the Build menu is clicked, as shown in Fig. 5.6, provides options for locking different parts of the strategy logic and orders while building. The options make it possible to limit the build process to specific parts of strategies while leaving other parts unaltered. For example, if the long and short entry conditions are selected, the build process will evolve the exit conditions and the entry and exit orders while leaving the entry conditions as they are in the initial population.

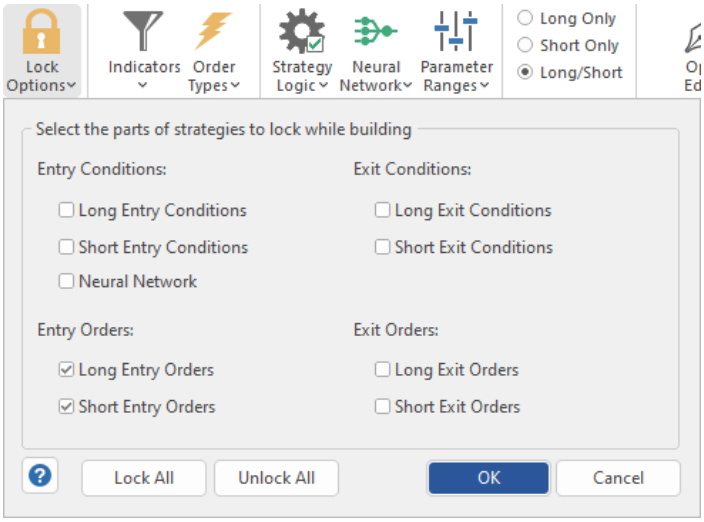


Figure 5.6. Strategy lock options.

Entry conditions refer to the logical rules for entry conditions evolved during building, which are assigned to the code variables EntCondL and EntCondS. Exit conditions refer to the logical rules used in conjunction with the "Exit at market" order type and assigned to the code variables ExCondL and ExCondS. Entry orders refers to both the type of order (market, stop, or limit) and the price (if not a market order). Exit orders refers to all current exit orders, including all related parameters and their values.

There are two ways the lock options would typically be used. If the population contains a strategy with elements you want to keep, you could use it to define the initial population and build from there. Alternatively, you could use the editor to define a strategy with the elements you want and let the build process determine the others. In either case, you would typically remove all other members of the population and clone the strategy to fill in the

initial population. You would use the lock options to select the strategy elements you want to keep, and the build process would determine the rest.

Some points to keep in mind when using the lock options:

- The lock options apply to all members of the population. It's not possible to lock specific strategies in the population while leaving others unlocked.
- Because the build process normally starts by initializing the population, which would erase the strategies with features you want to retain, the "Reset on Build" option under GP Settings should be deselected (off). If "Reset on Build" has been selected and lock options are present, a warning message will be displayed when the build button is clicked.
- If the symmetry option has been selected on the Strategy Logic menu, it will take precedence over the lock options. For example, if you lock the long entry conditions but not the short entry conditions and "Long/Short Symmetry" has been selected, the short entry conditions will be chosen based on symmetry.
- The lock options apply to the strategy cloning feature. This ensures that when the strategy you want to apply the lock options to is cloned to define the initial population, the elements you want to lock are retained throughout the initial population.

Indicators and Order Types

Indicators are selected on the Indicators drop-down window, shown in Fig. 2.7 in Quick Start Steps. The available order types are listed on the Order Types drop-down window, as shown in Fig. 2.8. The tables on these two windows represent the *build set*, which contains the indicators and 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 row in the Consider column of the table. To restore the previous settings, click the Reset button. To include all items for consideration during the build process, click the "Consider All" button, which will place an "X" in the Consider column for each item. To remove all items from consideration, click the "Consider None" 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 Entry and Exit Conditions, Introduction chapter. The indicators themselves are described in the appendix. The different types of entry and exit orders were discussed in Order Types, Introduction chapter.

Both windows contain 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 one of the three types of Protective Stop orders. If you select multiple protective stop orders in the Include column, the program will randomly select one of them to include in each strategy. For example, if you "include" both "Protective Stop ($)" and "Protective Stop (Price)", one or the other will be randomly selected for each strategy; both cannot be included because only one protective stop can be included in a strategy. The same functionality is provided for the three types of target orders and for the two types of trailing stops.

Checking the Include column for an indicator means the build process will attempt to include the indicator in each logical condition (long entry, short entry, long exit (if applicable), short exit (if applicable)), and in at least one neural network input (if present). If one of the order price indicators is included and price-based orders are used, the included indicator will also be used as the first price in the order. As explained in the introduction, the evolutionary

algorithm of Builder forms entry and exit conditions by constructing a tree of logical conditions consisting of the indicators. Because this process involves random selection, the Include column for indicators cannot guarantee that all included indicators will be part of each generated strategy.

If too many indicators are marked in the Include column, particularly if they have different return types, the algorithm may not be able to construct a rule tree that contains all the included indicators. If this happens, **the error message "Problem constructing rule tree" will be written to the Messages window. This message is informational only**; the process will be repeated until a valid rule tree has been constructed or the build process fails and stops. In some cases, other informational messages will be displayed in the Messages window to inform you that some members of the population may not contain all the selected indicators.

Note: If an indicator or order type is not available for the chosen trading platform, as determined from the code type selection on the Evaluation menu, the table entry will be grayed-out and will not be available for selection.

Strategy Logic Options

Options affecting the types of strategies generated by Builder can be made on the Trading Logic Options window, which drops down when the Strategy Logic button on the Build menu is clicked, as shown in Fig. 5.7.

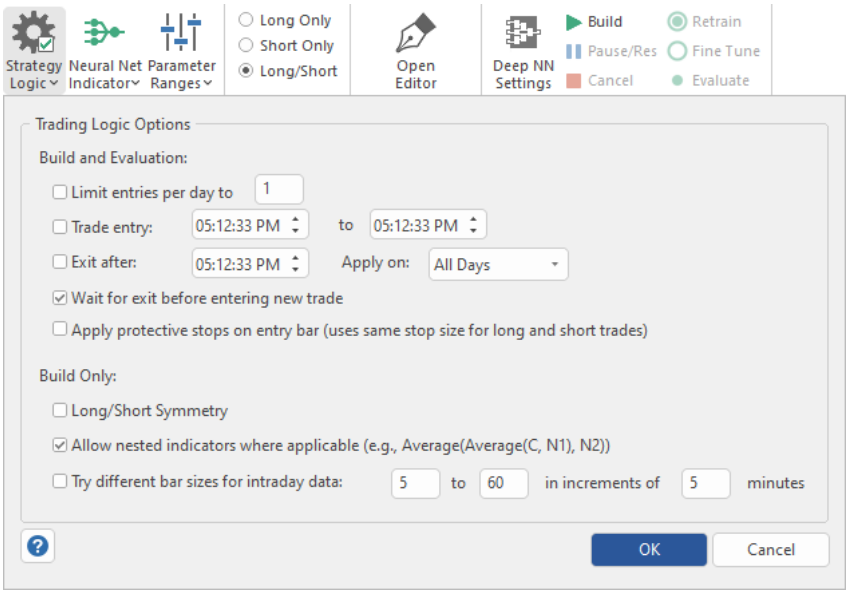


Figure 5.7. Trading Logic Options window on the Build menu.

The first group of options under the heading "Build and Evaluation" apply both when building and evaluating trading strategies. If any of these options are changed and a strategy is re-evaluated by clicking the Evaluate button on the Evaluation menu, the strategy logic and code will be modified accordingly and the performance results recomputed. The options under the heading "Build Only" only apply when building strategies and cannot be used to change a strategy during subsequent evaluations.

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.

Entry times for intraday trades can be restricted to a time range using the **Trade entry** option. Enter the starting and ending times using the time selectors. This will restrict entries to the time range chosen. To force trades to exit by a certain time, enter the desired exit time in the **Exit after** option. If an intraday trade is open at that time, it will be closed at market on the next bar. You can also specify the day of the week on which the exit-after option will apply using the drop-down menu next to “Apply on”. For example, select "Friday" to exit after the specified time only on Fridays. To apply the option on every day, select the option "All Days", which is the default.

Please note: The **Exit after** option uses the "Exit at Time" order type, as shown on the Order Types tab. Consequently, selecting this option also selects the "Exit at Time" order type. If you subsequently de-select this order type on the Order Types tab, you will have to select the **Exit after** option again to re-establish it.

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. Uncheck this option to create "stop and reverse" strategies that are always in the market, either long or short.

The option to **Apply protective stops on entry bar** immediately applies the protective stop order upon trade entry. This means the protective stop will be applied on the bar of entry, rather than waiting until the close of the bar like most orders. In TradeStation, selecting this option applies the protective stop order using the SetStopLoss EasyLanguage command. This option only applies to existing protective stop orders; it does not add a protective stop if one does not already exist. Also, because this option must accommodate the SetStopLoss command in EasyLanguage, it uses the same size stop for both long and short trades.

The **Long/Short Symmetry** option, if checked, constructs short-side conditions and orders by logically reversing the conditions and orders 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. This option supersedes the lock options, if any, in cases where they conflict.

Indicator nesting can be turned on or off using the option **Allow nested indicators where applicable**. If checked, this option allows indicators to take other indicators as input, such as Momentum(TriAverage(L, N1), N2). The number of levels of nesting is determined by the tree depth, which is determined for each strategy during the build process. Indicator nesting is not available for MetaTrader 4 strategies. In this case, the option is ignored.

The intraday bar size can be included as a parameter in the build process by selecting the option **Try different bar sizes for intraday data**. Provided the price data for the selected market support intraday bars, this option will temporarily set the data type to Intraday during the build process and vary the bar size between the range of values chosen as part of this option. For example, you might enter a bar size range of 5 to 60 in increments of 5 minutes. Builder will try different bar sizes ranging from 5-minute bars to 60-minute bars in increments of 5 minutes (i.e., 5, 10, 15, 20, …, 60). In this case, the selected price file in the Market Symbols table should contain 5-minute price bars or smaller. For example, 1-minute bars would also work well in this example, but 3-minute bars would not give accurate results for some of the bar sizes, such as 10 and 20-minute bars.

Builder will also permit this option if the underlying data are range or tick bars, although the resulting intraday bars may not be accurate in this case. The most versatile type of price data to use with this option is 1-minute bars, although larger minute bars are fine, provided the selected bar size range and increment result in bars that are an even multiple of the bar size in the price file. To ensure this is the case, the bar size range values and increment entered as part of this option should be integer multiples of the bar size in the price file. For example, if the price file contains 4-minute bars, the range and size increment should be multiples of 4, such as 4 to 40 in increments of 4.

When the intraday bar size is part of the build process, the bar size is evolved as part of the strategy, just like any other strategy feature or parameter. The resulting optimal bar size is reported in the build report under Strategy Description, Other Strategy Features, as well as in the Build Results table ("Opt Bar Size"). The reported results will be based on this optimal bar size. This will be true even if the currently selected bar type in the Market Symbols table is other than intraday or if the intraday bar size in the Market Symbols table is different than the optimal value. When selecting "Evaluate" or "Evaluate All" from the Evaluation menu, a strategy with an optimal bar size will be evaluated at that bar size only if the option "Use optimal bar size if applicable" is selected on the Evaluation menu. If this option is unchecked, the strategy will be evaluated using the settings currently in effect in the Market Symbols table, which may include a different bar type and/or bar size than the optimal one found during the build process.

Neural Network Indicator Settings

A neural network indicator can be included in each strategy by selecting the option **Include neural network indicator in entry conditions** on the Neural Network Indicator Settings drop-down window in the Build menu, as shown in Fig. 5.8. As explained in the section Neural Network Indicators, a neural network is a nonlinear function of one or more inputs, which Builder will select and evolve along with the entry and exit conditions if this option is included. Note that this is much different than the deep neural networks, which are much more complex and potentially much more powerful, that are discussed in Deep Neural Networks.

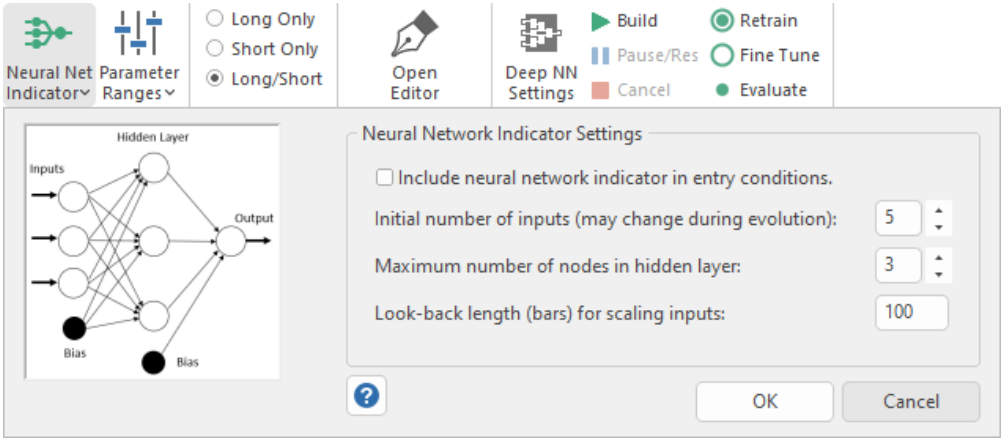


Figure 5.8. Available settings for the optional neural network indicator feature.

The output of the neural network indicator provides an additional filter for trade entries. To include a neural network indicator, check the box and enter the settings for the initial number of nodes, maximum number of nodes in the hidden layer, and the look-back length, as shown in Fig. 5.8. Please refer to Neural Network Indicators for more information.

Parameter Ranges

The Parameter Ranges drop-down window of the Build menu, shown in Fig. 5.9, allows you to optionally set the minimum and maximum values for constants used by the program, such as the sizes of the protective stops, 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 fixed exit stops that are too large, you can reduce the upper value for the **Fixed Stop Size, Exits**.

The ranges for fixed size stop and limit orders, both for entry and exit, are given in the currency in which the account is denominated and are specified per share or contract. For example, for US dollar-denominated accounts, entering a range of 100 to 1000 for **Fixed Stop Size, Exits**, would mean that if you had a fixed size exit stop in a strategy, the stop size would be somewhere in the range from $100 per contract to $1000 per contract. For stock and forex symbols, the entered values would typically be much smaller in order to be reasonable on a per-share basis.

Click the **Use Defaults** button to set the values to a default range calculated by the program for the currently selected market(s). The default range is calculated automatically based on multiples of the average true range (ATR). The lower limit is based on a multiple of 0.25 of the ATR, rounded down to the nearest tick, and the upper limit is based on a multiple of 4.0 of the ATR, rounded down to the nearest tick. If multiple markets have been selected, the range is based on the current selection, which is the market last clicked on. To see if the range changes significantly for other markets, click on a different market in the Market Symbols table and click the Use Defaults button.

Several order types use percentage values. The ranges **Percentage Entry Size, Stops/Limits**; **Percentage Stop Size, Exits**; and **Percentage Target Size** provide the values for these order types. The entered values are in percent. For example, if the range for Percentage Stop Size, Exits is set to 0.5 to 2.0, the size of the protective stop could be anywhere from 0.5% to 2.0% below the entry price for a long trade.

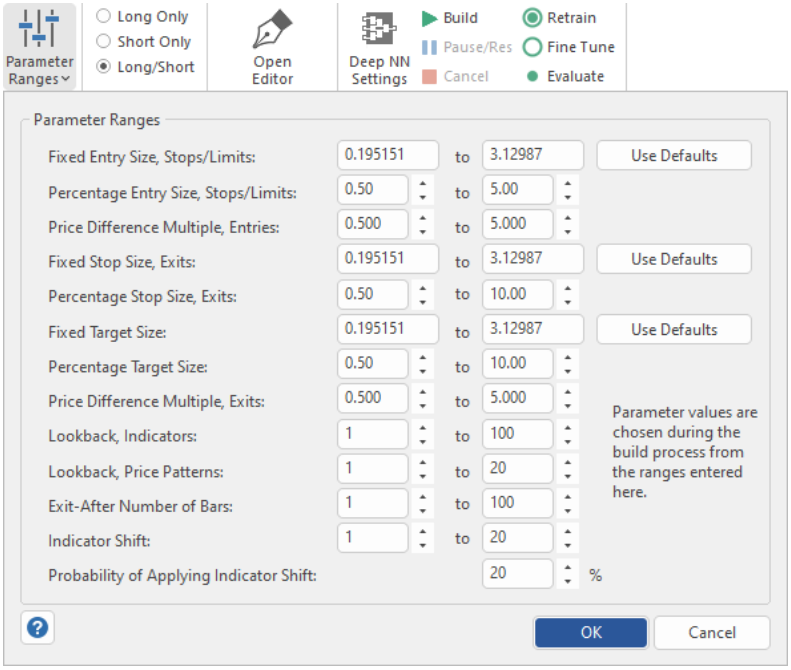


Figure 5.9. The Parameter Ranges drop-down window of the Build menu.

The range of values for **Price Difference Multiple, Entries/Exits** is used when constructing price-based stop and limit orders for both entry and exit orders and for trailing stops with an ATR floor. This price difference multiple is multiplied by the price difference, including ATR and true range, before adding it to a reference price to calculate the order price.

The parameter range for **Lookback, Indicators** applies to all indicators requiring a lookback length except for price patterns; e.g., the averaging length of a moving average.

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

The number of bars for the order types that exit after a fixed number of bars, including exiting if profitable and exiting at a loss, is selected from the parameter range **Exit-After Number of Bars**.

Whether or not an indicator is shifted depends on the **Probability of Applying Indicator Shift**. The shift value itself is chosen from the parameter range **Indicator Shift**. A shifted indicator returns the value N bars ago, where N is the shift. For example, in EasyLanguage, XAverage(C, 14)[N] returns the value of the exponential moving average N bars ago. If the probability is set to 20%, for example, there is a 20% chance that any indicator that can be shifted will be shifted. Set the probability to 0% to turn indicator shifting off.

The different types of orders and which parameter ranges they use are discussed in Order Types in the Introduction chapter.

Adaptrade Editor

The next option on the Build menu is "Open Editor", which opens Adaptrade Editor ("Editor"), a point-and-click trading strategy editor. There are several different ways to open Editor. Clicking the **Open Editor** button opens Editor with a blank/empty strategy. Alternatively, you can right-click a strategy in the Build Results table and select "Edit Strategy" or select "Edit Strategy" from the File menu, either of which will open the editor with the strategy that's currently selected.

The editor allows you to modify strategies generated by Builder and/or create new strategies from scratch. To create a new strategy and add it to the existing population, click the Open Editor button in the Build menu and use the editor to create your strategy. The editor allows you to specify logical conditions for entry and exit, enter parameter values, select order types, and make other settings that define the strategy. Once the strategy has been created, save it from the File menu of Editor, which will save it to a file with the extension .gpcode (e.g., MyNewStrategy.gpcode). You can then use the Import function of Builder (accessed via the File menu or by right-clicking the Build Results tables) to add the strategy to either the main or Top Strategies population.

To edit a strategy generated by Builder, select the strategy in the Build Results table and either select the edit command from the File menu or from the right-click (context) menu of the Build Results table. This will open the strategy in the editor using a temporary strategy file. You'll be able to change parameter values, modify logic, and make any other changes allowed by the editor. To save the changes, click Save via the File menu in the editor, which will update the strategy in Builder. To see the effects of the changes, click Evaluate from the Evaluation menu in Builder to run the back-test on the edited strategy. When the editor window is closed, the temporary file will be deleted automatically.

When working with DNNs, the editor can be used to edit the inputs to the DNN. However, it's not possible to add a DNN to a strategy via the editor.

Note: If a strategy built using the "Long/Short Symmetry" option is opened in the editor, the inputs, which are shared for the long and short sides in Builder, will be repeated for each side of the market in the editor. This allows you to change the strategy elements for each side of the market separately in the editor. The duplicated inputs will be retained in Builder when the strategy is saved in the editor.

Editor comes with its own user's guide, which can be accessed by pressing the F1 key from within the program (or download the PDF version from adaptrade.com). Please refer to the user's guide for Editor for specific instructions on using the program.

Note: While Adaptrade Editor is a stand-alone Windows program, if it's installed as part of Adaptrade Builder, it can only be opened from within Builder and used while Builder is open.

Deep Neural Network Settings

As explained in Deep Neural Networks in the Introduction, most of the settings for building populations of deep neural networks (DNNs) are made on the DNN Settings window, shown below in Fig. 5.10.

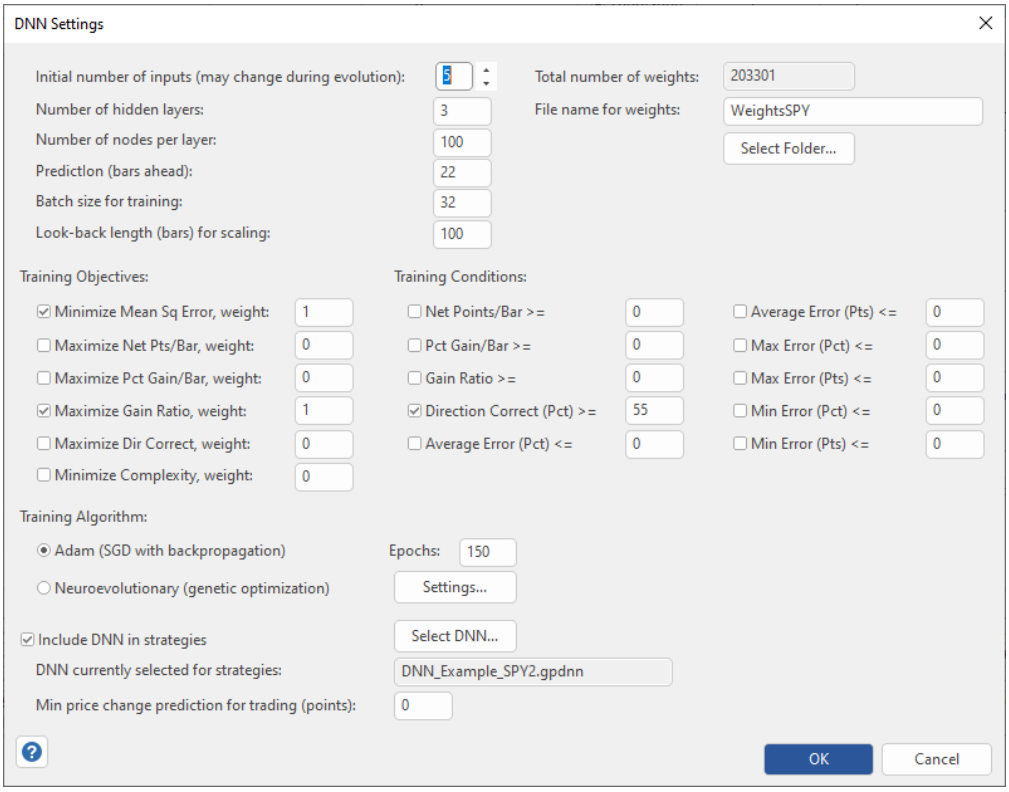


Figure 5.10. The DNN Settings window of the Build menu is where the settings are made for building and using deep neural networks.

The first group of settings define the architecture of the DNN. This includes the initial number of inputs to the DNN, which can be typed in the box or incremented or decremented using the buttons next to the edit box. The inputs to the DNN consist of technical indicators

that are drawn from the available indicator types during the evolutionary (genetic programming) build process. The purpose of the build process is to find the set of inputs/indicators that generates the best results, as defined by the training objectives and conditions, discussed below. Because the build process modifies the inputs at each generation, the actual number of inputs to the DNN may be different at the end of the build than the initial number entered here.

The other elements that define the DNN architecture are the number of hidden layers and the number of nodes per layer. The number of hidden layers defaults to a value of three, and the number of nodes per layer defaults to 150. Smaller values will result in a faster build process but generate DNNs that may be less accurate. The total number of weights (parameters) in the DNN, which is determined by its architecture, is shown to the right of the inputs. The weight values are set by the training process, which is performed after the inputs are selected by the GP process at each generation.

You can also choose the prediction length in bars. This is the number of bars that the DNN will be trained to predict forward. For example, a prediction length of 5 means the DNN will be trained to predict the closing price 5 bars ahead.

In general, smaller prediction lengths will provide lower overall prediction errors (mean squared errors) but lower accuracies in the prediction direction. For example, with a prediction length of 1, it will be difficult to train the DNN to predict the direction of the next close -- up or down relative to the current close -- with an accuracy much greater than 50% due to the mostly random nature of bar-to-bar price changes. However, the correspondence between the predicted prices from the DNN and the actual market prices will be relatively good. On the other hand, with a prediction length of, say, 30, the accuracy of the direction prediction may be substantially greater than 50%, but the correspondence between the predicted prices and the actual prices will be much lower than with a smaller prediction length.

You can optionally select the batch size for training and the look-back length for scaling the input values. The batch size is the number of bars of data used for calculating the metrics during training. The specific bars are randomly chosen from the training segment. The default value for the batch size is 32, which is typical for training DNNs. Smaller batch sizes will result in faster build times but increase the variability in results, which can negatively affect convergence. The look-back length defaults to 100. This is the length over which the input values are scaled so that they lie within the range 0 to 1.

The weights for each DNN in the population are stored in separate files where the name of each file is based on the name entered on this window ("File name for weights") followed by its member number in the population. For example, if you enter the name "MyWeights", the weight file names will be "MyWeights_1.nnwgts", "MyWeights_2.nnwgts", and so on, where "nnwgts" is the file extension that Builder uses for DNN weight files. You can also select the folder where the weights files will be saved by clicking the "Select Folder" button. The default folder location is in your documents folder in Adaptrade Builder\Weights\. It's recommended that you create a new folder within the Weights folder for each build. To do so, click the folder icon in the upper right-hand corner of the Select Folder window. If you don't select a folder, Builder will create one using the current project name.

Note: The weight files for a DNN are saved when the project file is saved (via either Save or Save As from the File menu) or when a DNN is saved to a separate file (see below). If a DNN is copied to the Top Strategies population, its weight file will be assigned a unique file name to avoid conflicts with any member of the main population that has the same member ID. This can result in multiple copies of the weight file if the file is saved repeatedly.

Although it’s not necessary to do so, you can refer to the Build Report for the name of the weight file being used and delete any extraneous weight files if desired.

The next two groups of settings are for the training objectives and conditions, which define the fitness for evolving the population of DNNs and for training each DNN. The objectives and conditions on this window work much like the build objectives and conditions on the Metrics window. A linear combination of the objectives is formed, where each one is multiplied by the entered weight values. The training conditions form penalty terms for any condition that's not met. It's recommended that the mean squared error metric always be selected. As with building strategies, other objectives and conditions can be selected based on the initial build results and your specific goals and preferences. For a description of each metric, please refer to the appendix Performance Metrics; the DNN metrics are listed at the end.

A few points are worth noting:

1. Unlike the metrics used when building strategies, the metrics shown here are not scaled when forming the fitness. This means metrics with large values, such as the Percent Dir metric, may contribute more to the fitness than a metric with a smaller value, such as Mean Squared Error.
2. The same metrics are used both for evolving the DNN inputs via the GP algorithm and for training each DNN using the training method discussed below.
3. The training error is defined as the negative of the fitness since the training process minimizes the training error whereas the GP process maximizes the fitness. A consequence of this is that although the DNN training convergence plot displays the "error", it's possible that the error value can be less than zero since it's defined as the negative of fitness and fitness can be a positive number.
4. When building and training DNNs, the build metrics of the Metrics window are not applicable and are ignored.

There are two choices for the training algorithm, which determines how the weight values for each DNN are calculated. The first option is stochastic gradient descent (SGD) with backpropagation, implemented via the Adam variant. This is a classic technique used to train DNNs. It's relatively fast and usually works well. If Mean Sq Error is the only metric, you will most likely see a convergence plot in which the training error declines relatively smoothly and steadily. However, for some combinations of metrics, convergence can be erratic. The alternative is the neuroevolutionary method, which is implemented as a simple genetic algorithm (GA). While usually much slower and more memory intensive than SGD, the GA algorithm by design guarantees that at each step, the error is no worse than at the preceding step, whereas the error with SGD can increase at some steps, depending on the topography of the search space.

If selecting the SGD algorithm, enter the number of epochs in the space provided. This is the number of training steps, where at each step, the gradient of the training error is calculated, and the weight values are updated. If selecting the GA algorithm, click the Settings button and enter values for the population size, number of generations, crossover percentage, and mutation percentage.

The final settings on this window are for including a DNN in trading strategies. After a population of DNNs has been built, it's necessary to save any that you want to include in strategies. As explained in the introduction, you do this by right-clicking the DNN entry in the Build Results table and selecting the option to "Save Deep Neural Networks". Assuming you've already done this, you can include one of the saved DNNs in a subsequent strategy build by checking the box "Include DNN in strategies" and clicking the button "Select

DNN". This will open the window shown below in Fig. 5.11 for selecting the saved DNN you want to include.

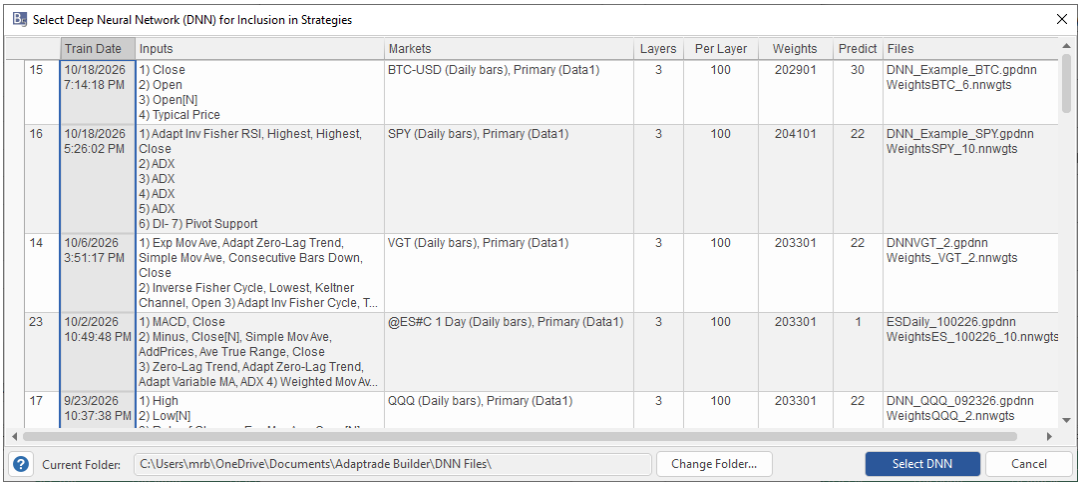


Figure 5.11. The DNN selection window allows you to select a saved DNN to be included in strategies during the strategy building process.

The indicators chosen by the DNN build process as inputs are listed in the Inputs column. Other columns show the market(s) on which the DNN was trained, the number of layers and nodes per layer, as well as the length of the prediction window ("Predict") and the names of the DNN and weight files. You can click on any column heading in the table to sort the table by that column. Click any row to select the corresponding DNN and click the Select DNN button to confirm your selection. The window displays the DNNs it finds in the folder shown at the bottom of the window ("Current Folder"). To choose a DNN located elsewhere, click the "Change Folder" button. After closing the window, the selected DNN will be displayed on the DNN Settings window in the entry for "DNN currently selected for strategies".

The final option on the DNN Settings window is the "Min price change prediction for trading (points)". This is the minimum number of points above (for a long trade) or below (for a short trade) the current close that the DNN predicts the market will close at for a trade to be entered. Specifically, for a long trade, including a DNN in a strategy adds the following condition: DNNPrice - C > MinChange, where DNNPrice is the DNN's predicted close, C is the current close, and MinChange is the minimum price change prediction. For a short trade, it adds the condition: C - DNNPrice > MinChange. A trade is entered if this condition is met along with any other entry conditions. The default value for this option is zero, which implies the DNN entry condition would be met if the DNN prediction is above the current close for a long entry or below it for a short entry.

Market Prediction Panel

The Market Prediction Panel of the Build menu, shown below in Fig. 5.12, contains the commands that control the build process for DNNs.

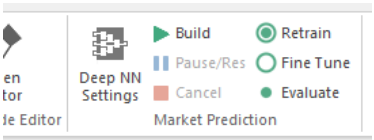


Figure 5.12. The Market Prediction Panel of the Build menu contains the commands for controlling the process for building populations of DNNs.

The Build button starts the build process, in which the technical indicators that form the inputs to the DNNs are evolved using a genetic programming algorithm over successive generations. The GP process is guided by the settings on the GP Settings window as well as by the DNN metrics discussed above and the other build settings. To pause the build process without cancelling it, click the Pause/Res button. Click the button again to resume the build. The build process can be stopped by clicking the Cancel button. Provided at least the initial population has been built, the results will be retained and displayed in the Build Results table, and the weight values for each DNN will be retained until they're saved in the files specified by the weight file name chosen on the DNN Settings window.

The Retrain and Fine Tune buttons can be used to modify the weights of a DNN. Both commands train the DNN using the training algorithm selected on the DNN Settings window -- either the SGD or genetic algorithm. Retraining starts the training process from a set of randomly chosen weight values, whereas fine tuning starts the training from the current weight values. Fine tuning essentially continues the training from where the build process ended, whereas retraining starts over.

Retraining can be used to try different training methods or settings for the same DNN inputs. You can also change the architecture of the DNN -- specifically the number of nodes per layer or number of layers -- and retrain. Changing the setting for the initial number of inputs will have no effect since the inputs can only be changed by the build process or via the editor. The prediction length, batch size, and look-back length can also be changed when retraining or fine tuning, as can the weight file name and folder. If the DNN's architecture is changed prior to fine tuning, the weight values will be re-initialized, as with retraining. One application of fine tuning is to try the other training algorithm. For example, if SGD was used during the build, the DNN can be fine tuned using the genetic algorithm to see if it provides further improvement.

For additional information and discussion about using Builder to build deep neural networks, please see the Deep Neural Networks section of the Introduction.

Evaluation Options

The Evaluation menu is shown in Fig. 2.10 in Quick Start Steps in the Getting Started chapter. This menu contains options that are applied when a strategy is evaluated, which occurs during strategy building and when the Evaluate or Evaluate All commands are selected. In this context, "evaluation" means performing a back-test on the selected strategy, which simulates trading the strategy over the markets selected in the Market Symbols table on the Build Symbols or Evaluation Symbols window.

Unlike most of the settings on the Build menu, which affect how the strategies are built but can't change them after they're built, the options on this menu are applied whenever a strategy is evaluated. As a result, you can change the options on this tab, such as an order fill rule or the code output option and re-evaluate an existing strategy to see the effect. **Re-evaluating a strategy recalculates the performance results and rewrites the code, accounting for any changes to the options and settings discussed here and in Strategy Logic Options.**

Evaluate Strategies Panel

This panel of the Evaluation menu contains the Evaluate and Evaluate All buttons. Clicking the Evaluate button will perform a back-test of the strategy currently selected in the Build Results table. A message will be displayed in the Messages window when the evaluation is complete, and the results windows will be updated to display the results. Clicking the Evaluate All button will evaluate all the strategies in the population. The progress of the

evaluation will be displayed in the Messages window. The process can be cancelled by clicking the Cancel button. The evaluation is performed on the symbols currently selected on the Evaluation Symbols window.

Simulation Settings Panel
This panel contains settings that affect the back-test results of the evaluation. The **Starting Equity** is the value of the account equity at the start of the back-test. Depending on the option selected on the Position Sizing menu for "Equity Calculation", the simulated account equity is either accrued during the back-test or set equal to the starting equity for each trade. In either case, the starting equity affects the calculation of any performance metric that depends on the account equity.

The option **Apply stress tests/Monte Carlo analysis** optionally evaluates the strategy results using the stress testing and Monte Carlo options on the Stress Testing window; see the section Stress Testing and Monte Carlo Analysis under Usage Topics for a more detailed explanation of the stress testing process. Briefly, stress testing involves evaluating the strategy multiple times, each time under slightly different conditions; for example, using randomized prices or randomized input parameter values. When stress testing, the different evaluations are combined using Monte Carlo analysis, and the results are reported at the confidence level specified on the Stress Testing window.

The option **Use optimal bar size if applicable** works in conjunction with the option on the Strategy Logic window to try different bar sizes for intraday data when building. If that option is selected, the option here will evaluate the strategy using the optimal intraday bar size found during the build process. On the other hand, if the option in this section is unchecked, the strategy will be evaluated using whichever market data settings are currently in place in the Market Symbols table on the Evaluation Symbols window. This option has no effect unless the strategy being evaluated has an optimal intraday bar size found from using the option to try different intraday bar sizes.

The other options in this panel are accessed through the Order Placing Options drop-down window, shown below in Fig. 5.13.

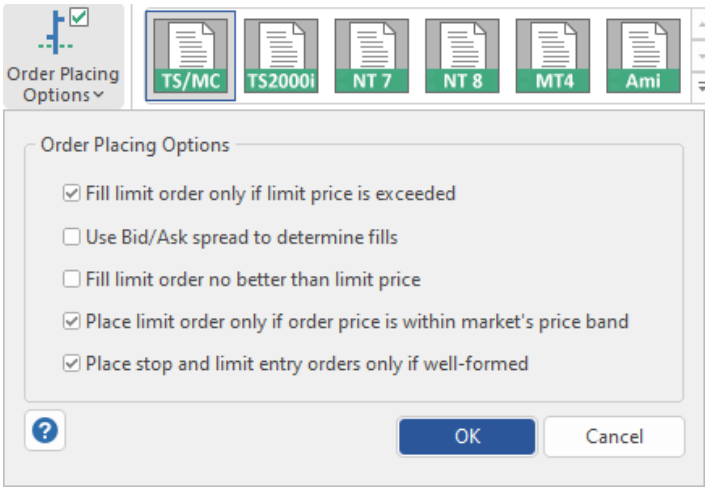


Figure 5.13. The Order Placing Options drop-down window of the Evaluation menu.

The option **Fill limit order when limit price is exceeded** only fills limit orders when the limit price is exceeded. If unchecked, limit orders will be shown as filled when the limit price is touched. For strategies applied to small bars, such as 1-minute or tick bars, it will usually

be more accurate to assume that a limit order is not filled unless the limit price is exceeded. In practice, a limit order may not be filled if the limit price is only touched because there may be insufficient volume to fill all the orders at that price or better. Therefore, it's prudent to select this option when the volume on a typical price bar is expected to be small.

The option **Use Bid/Ask spread to determine fills** uses the bid/ask spread to determine fill prices and whether or not stop and limit orders are filled. The default setting is "off". Since MetaTrader 4 uses the bid/ask spread in this manner, this option should always be selected when building and evaluating strategies for MetaTrader 4. Please see Symbol Settings in the Symbols and Settings chapter for more information on the bid/ask spread.

The option **Fill limit order no better than limit price** prevents a limit order from being filled at a more favorable price due to a gap opening. For example, if an order to buy on a limit is in effect and the next bar opens below the limit price, it's possible that you would get filled at or near the open of the bar, which would be a more favorable fill than the order price. However, it's also possible that the order would not be filled until the market went back up to your order price. This option assumes the worst-case scenario of being filled at no better than the order price. Selecting this option will provide conservative fill estimates for limit orders.

If the symbol is a futures contract traded through the Chicago Mercantile Exchange (CME), limit orders may be subject to price banding. The option **Place limit order only if order price is within market's price band** means that the limit order will only be accepted by the exchange if the order price is within the band size of where the market is currently trading. The price band size is a symbol property; see Symbol Settings in the chapter Symbols and Settings for more information. Select this option if the selected symbol is subject to price banding for limit orders and be sure to provide a valid value for the price band size in the symbol's properties.

The option **Place stop and limit entry orders only if well-formed** only places a stop or limit entry order if the market has not moved beyond the order price. For long stop and short limit orders, this means the market must be below the order price at the time the order is placed (i.e., on the close of the prior bar). For short stop and long limit orders, the market must be above the order price on the close of the prior bar. If this option is not checked, the stop or limit entry order will be placed regardless of where the market is trading relative to the order price. In this case, if the market is beyond the order price, the order will be automatically converted into a market order. Note that dollar-based and percentage-based stop and limit entry orders are always well-formed because of how the entry prices are constructed. As a result, this option is only pertinent to price-based stop and limit entries.

Code Type Panel

The code type panel allows you to choose the type of code generated when the strategy is evaluated. Select TS/MC for TradeStation version 6 and above and for recent versions of MultiCharts, TS 2000i for TradeStation version 2000i, NT 7 for NinjaScript code for NinjaTrader 7, NT 8 for NinjaScript code for NinjaTrader 8, MT4 for MQL4 code for MetaTrader 4, or Ami for AFL code for AmiBroker. The code for all of the saved strategies listed in the Build Results table is stored in the project file (extension .gpstrat). If you want to generate code for a different platform, change the selection here, select the desired strategy from the Build Results table, and select Evaluate from this menu. The code will be re-written for the new code type as part of the evaluation process.

Note: As shown in Table 1 in the Introduction chapter, not all indicators are available for all code types. Also, some indicators give different results in each platform. For this reason, the same results may not be found if the code type is changed and the strategy is re-evaluated.

Also, if an indicator is not available for one platform, converting to a different code type may result in invalid code. This can also happen due to indicator nesting, which is unavailable in MetaTrader 4. Invalid code may also result if a strategy with both long and short trades is converted to AmiBroker code; Builder does not support both long and short trades in the same strategy for AmiBroker code.

Significance Test Panel
This panel allows you to perform a unique test of statistical significance that's designed to uncover when a strategy's apparently good performance is, in fact, the result of good luck rather than effective trading logic. This test should not be confused with the so-called "Trade Significance" metric that's part of the set of performance metrics calculated by Builder. The Trade Significance metric (aka "Significance") performs a traditional *t* test on the average trade for a strategy. This can provide a measure of strategy quality but cannot assess whether the results for the strategy are statistically significant because such a test does not take the data mining process into account.

The significance test in this panel takes the build process into account to construct a sampling distribution for each strategy in the population. The sampling distribution is based on the total number of strategies constructed as part of the build process, including any resets due to the Build Failure Rules. For example, if 10,000 strategies were generated before arriving at the final population, then each point on the sampling distribution will be produced by generating 10,000 random strategies and taking the best one. This process is repeated M times, where M is the number of samples.

The randomly generated strategies have no predictive power. However, the best one among a sample of 10,000 is likely to generate good results simply due to luck. We want to know if the strategy in the population is significantly better than the best random strategy, so the sampling distribution is constructed from M strategies, each of which is the best strategy among all the (e.g., 10,000) randomly generated strategies. Please refer to the article **Is That Back-Test Result Good or Just Lucky?** under the Newsletter link at www.Adaptrade.com for more information.

The **Number of Samples** entry is the number of samples comprising the sampling distribution; e.g., M in the example above. The option **Run test after each build** will cause this test to be run automatically at the conclusion of each build. If this option is not selected, you can manually run the test at any time by clicking the button **Perform Significance Test**. Builder stores the total number of strategies generated during the build process, which it will use in constructing the sampling distribution when the test is run later. *If you run the test after building, the same data used for the build must be available; otherwise, the test will not run.*

The test is applied to each strategy in the population. The strategy's fitness, *as evaluated over all segments combined*, is used as the test statistic. The results of the test are displayed in the Build Results table, listed under "Build Sig", in the right-most column in the table. The result is listed as the probability value (0 - 1) with a Pass/Fail label in parentheses; for example, "0.961 (Pass)". A significance level of 0.95 (95%) is the threshold for passing the test. In other words, a probability value of 0.95 or greater is considered statistically significant.

Please note: The test is very time-consuming. Each sample requires numerous strategies to be randomly generated, similar to generating a very large initial population. For example, if there are 10,000 strategies in each sample and 500 samples, a total of 5,000,000 strategies must be generated to construct the sampling distribution. If the test is taking too long, it can be cancelled by clicking the Cancel button.

Position Sizing Settings

The Position Sizing menu is shown in Fig. 2.11 in Quick Start Steps in the Getting Started chapter. Position sizing determines how many shares or contracts are traded on each trade. The position sizing method used when evaluating a strategy is also implemented in the strategy code generated by Builder, so that when running the code in the trading platform, the same position sizing should be seen as in Builder.

As with the options on the Evaluation menu, changes to the position sizing can be made after the strategy is built. To see how different position sizing settings affect the results, change the position sizing here and select Evaluate from the Evaluation menu. When the strategy is re-evaluated, the strategy code will be re-written to include the new position sizing settings.

Position Sizing Method Panel

Builder includes six different position sizing methods. Each method has an associated parameter value, which is entered in the **Parameter** field.

Fixed size. This method simply uses the value you enter as the number of shares or contracts for each trade. A fixed size of 1, for example, will trade one contract or share for each trade. For forex trading, a typical fixed size would be either 10000 or 100000. It's important to set the upper size limit to a value at least as big as the fixed size entered here.

Constant value. With constant value position sizing, each position is sized so that it has a specified value, such as $1000 per trade. This method can be used when you want to allocate a specified amount of equity to each trade. The constant value method will determine the number of shares or contracts corresponding to your specified amount. For example, if you plan to purchase a stock at a price of $25 and you want to spend $35,000, you would trade 35000/25 or 1400 shares. In this case, the constant value amount is 35000; each trade would be sized so that the position value is $35,000.

Percent of equity. In this method, the number of shares or contracts is chosen so that the value of the position is equal to the selected percent of account equity. For example, if the percent of equity is 40%, the position size will have a value equal to 40% of the account equity. If the account equity were $30,000, the position would have a value of 0.4 x 30000 or $12,000. If the share price were $25, the position size would be 12000/25 or 480 shares. In other words, a position size of 480 shares at $25 per share has a value of $12,000, which is 40% of the equity value of $30,000.

Fixed fractional. In fixed fractional position sizing, the number of shares or contracts is based on the risk of the trade. For example, you might risk 2% of account equity on each trade. Fixed fractional position sizing has been written about extensively by Ralph Vince. See, for example, his book "Portfolio Management Formulas," John Wiley & Sons, New York, 1990.

The risk of a trade is defined as the dollar amount that the trade would lose per contract or share if it were a loss. In Builder, the trade risk is taken as the size of the money management stop applied, if any, to each trade. If the strategy doesn’t use protective (money management) stops, the trade risk will be taken as the largest historical loss per contract or share. The largest loss is the largest loss up until the point at which the trade is taken. If subsequent losses are larger, the trade risk will be larger for subsequent trades.

As an example, consider a stock trading system that uses a 2-point stop. It might enter long at a price of 48 with a sell stop at 46. The risk per share would be $2. With a fixed fraction of 5% and account equity of $20,000, fixed fractional position sizing would risk 0.05 * 20000

or $1,000 on the next trade. Since the risk per share is $2, this means 1000/2 or 500 shares would be traded.

Fixed ratio. In fixed ratio position sizing the key parameter is the delta. This is the amount of profit per share or contract to increase the number of shares or contracts by one. A delta of $3,000, for example, means that if you're currently trading one contract, you would need to increase your account equity by $3,000 to start trading two contracts. Once you get to two contracts, you would need an additional profit of $6,000 to start trading three contracts. At three contracts, you would need an additional profit of $9,000 to start trading four contracts, and so on. Fixed ratio position sizing was developed by Ryan Jones in his book "The Trading Game," John Wiley & Sons, New York, 1999.

Normally, the delta is the amount per share or contract, as in the example above. For some instruments, such as forex, this would be inconvenient because forex uses a point value of 1 and typically trades in increments of 10,000 or 100,000 shares. To accommodate this, the delta is specified per multiples of shares when the option to round the share size to the nearest number of shares (see below) is selected. For example, if the "round to" option is selected with an increment of 10,000 shares (i.e., "Round to nearest 10,000 shares"), then the delta would be specified as the amount per 10,000 shares.

The fixed ratio method will start trading with a position size equal to the minimum number of shares/contracts specified in the Position Sizing Options panel (see below). For example, if the minimum number of shares for a forex strategy is set to 10,000, this will be the position size calculated by the fixed ratio method for a profit of zero. As profits are accrued, the position size will increase, based on the delta, in increments specified by the "round to" option.

Fixed amount per share. In this position sizing method, you choose the amount of account equity required to trade each share or contract. For example, if the dollar amount is $5,000, you would trade one contract for every $5,000 in the account. If the account equity is currently $50,000, the position size would be 10 contracts for the next trade.

The account equity value used in the position sizing calculations is determined by the option selected in the Equity Calculation panel, as explained below. For either option, the starting equity value, entered on the Evaluation menu, is the value of the simulated trading account at the start of trading. The default value of starting equity is $100,000.

Note: When building over multiple markets together (i.e., a portfolio), the same position sizing (including parameter values) is used for each market, and the account equity is the portfolio equity (i.e., the equity used to trade all markets in the portfolio together).

Recommendations: When starting a new build, it's often best to keep things simple. For that reason, when building strategies for the futures or forex markets, it's recommended to initially use the fixed size position sizing method. For stocks, the percent of equity method with a value of 100% is recommended (i.e., fully invested on each trade). Other methods can be selected and applied after a strategy is built by re-evaluating the strategy.

Position Sizing Limits Panel

In the Position Sizing Limits panel, you can enter the minimum and maximum number of shares or contracts that the strategy will trade by entering values for the **Size Limits** entries. The minimum value, which defaults to 1, is applied to all trades. After the position size is calculated, the position size is increased to the minimum if the position size would otherwise fall below the minimum. Likewise, the maximum, which defaults to 100, is applied to all

trades. After the position size is calculated, the position size is reduced to the maximum if the position size would otherwise exceed the maximum.

The option **Round to nearest** rounds the position size to the nearest number of shares or contracts. For example, if you enter a value of 100, the number of shares will be rounded to the nearest 100 shares. The default value is 1. For forex trading, this can be used to trade in even lot sizes of 10,000 or 100,000 by setting the rounding value to 10,000 or 100,000, respectively. In this case, the minimum number of shares should be set to the same value and the maximum value to some multiple of the minimum.

Evolve Position Sizing Panel
You can optionally choose to have Builder evolve the position sizing as part of the build process. There are three options. The first option, **Don't evolve; use selected settings**, uses the settings you specify for each strategy. If you select **Evolve parameter only**, the program will evolve only the parameter value (e.g., the fixed fraction) while using the selected position sizing method. If you select **Evolve method and parameter**, then Builder will evolve both the position sizing method (i.e., one of the six available methods) along with the associated parameter value.

Equity Calculation Panel
There are two options for instructing Builder how to calculate the account equity value used for position sizing and back-testing in general. If the option "Accumulate equity trade to trade" is selected, the program will track the account equity during the back-test from trade to trade, starting with the first date in the trading period, where the account equity is equal to the starting equity value, and accumulate the profits and losses up through the last date in the date range. As the equity changes, the position size will generally change as well. In all position sizing methods except for fixed size and constant value, the position size increases as the account equity rises and decreases when the account equity drops. For a portfolio of markets, the account equity for this option is the accumulated equity from the trade profits and losses from all markets in the portfolio.

If the option "Use entered equity value for each trade" is selected, the account equity for every trade is simply the starting account value entered on the Evaluation menu. In this case, the profits and losses are not accumulated from trade to trade. This option is useful when you want to see what the position size would be for a specific value of account equity, such as when using Builder as a trading platform. In this case, you can enter the current value of your account equity as the starting equity value and re-evaluate the strategy. The corresponding position size will be displayed in the Trading Orders window for the next (upcoming) trade if there is one. Likewise, if you're trading a Builder-generated strategy in your trading platform, you would most likely want to select this option because it will enable you to enter the current value of your account equity as a strategy input. The strategy would then generate the correct position size for the entered account value.

Reset Panel
To return to the position sizing that was selected or evolved with the strategy when it was built, click the **Reset to Build Settings** button then select Evaluate from the Evaluation menu to rewrite the code and re-calculate the results.

Chapter 6
Build Results

The results generated by running Builder include the strategy code for each of the saved strategies or deep neural networks (DNNs) along with the performance results for each one. The docking windows (panes) described below display the different types of program output. The Getting Started chapter discussed working with windows and panes in Builder.

Build Progress Window

When building trading strategies, the Build Progress window, shown in Fig. 6.1, has two plots, which display the progress of the build process at each generation. The Build Failure Tracking plot depicts the results of the Build Failure Rules defined on the GP Settings window (Build menu). The curve labeled "Ave Fit" is the average fitness of the population members on the test segment at each generation. The "Fit MA" curve is the moving average of the Ave Fit curve, where the look-back length is defined on GP Settings. The "Metric" curve shows how the average of the selected metric over the top N strategies on the test segment changes from generation to generation during the build process. Ideally, all three curves should slope upward to the right as the build progresses, as shown in Fig. 6.1, indicating improving performance on the test segment. The Build Failure Rules on GP Settings allow you to reset the build if this is not the case.

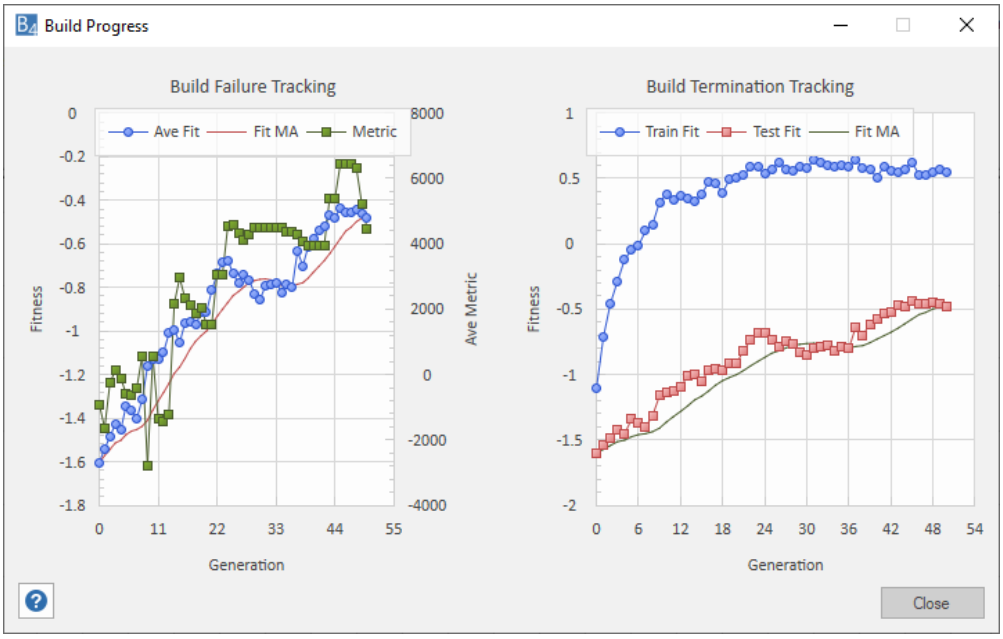


Figure 6.1. The Build Progress window displays and updates the Build Failure Tracking and Build Termination Tracking plots while strategies are being built.

The Build Termination Tracking plot depicts the build termination rules of the GP Settings window. The curve labeled "Train Fit" shows how the average population fitness on the training segment progresses from generation to generation. The "Test Fit" curve shows the

same results for the test segment. "Fit MA" is the moving average of the test fitness curve, as defined on GP Settings.

For a good quality build, the fitness on both the training and test segments should generally be increasing at successive generation, as shown in the figure. When the fitness starts to drop on the training segment, no further improvement should be expected in the strategies. More importantly, if the fitness on the test segment starts to level off or decline while the fitness on the training segment is still increasing, this can suggest that the strategies are being over-fit. In this case, the build process can be terminated, which is the purpose of the build termination rules on GP Settings. However, there are cases where the fitness can decline for some number of generations before increasing again. For this reason, it's often a good idea to run the build process for 100 generations or more before terminating it. Provided the option to save the generation with the highest fitness on the test segment has been selected (GP Settings window), there's no downside to this approach other than a longer build time.

When building DNNs, the Build Progress window only displays the Build Termination Tracking plot. The first build failure rule also applies to DNNs, but the second one involving a build metric does not since the strategy metrics are not used for building DNNs. As a result, the Build Failure Tracking plot would provide the same information as the curves for the test segment on the Build Termination Tracking plot.

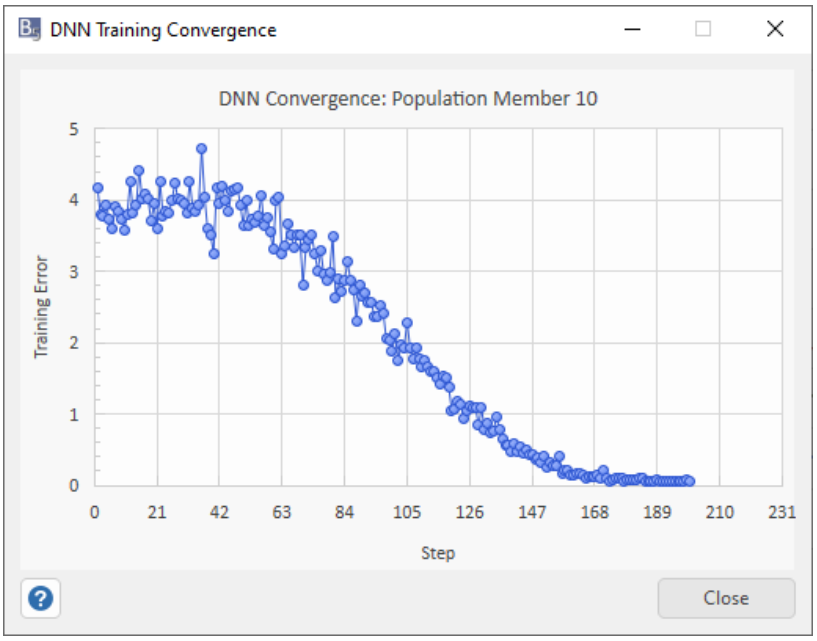


Figure 6.2. The DNN Training Convergence window displays and updates the training error for the currently selected DNN during the build process.

The program also displays a separate DNN Training Convergence window when building DNNs, as shown in Fig. 6.2. The DNN Training Convergence window shows how the training error changes during the DNN build process for the DNN currently selected in the Build Results table. You can select a different row in the Build Results table to view the convergence plot for the corresponding DNN. When the training error is equal to the Mean Squared Error metric and the training method is stochastic gradient descent, the training error will likely look like the plot in Fig. 6.2. The convergence may be more erratic when the training error is based on a combination of metrics. When using the genetic optimization training method, each step in the plot will be no higher than the prior one, although the error may not change for several steps in a row.

Messages Window

The Messages window displays messages generated by the program while performing build and evaluate operations, reading price files, and so on. 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. Other notable error messages are described below.

Error messages can either mean the build process was not successful or there was a recoverable error during an otherwise successful build. If the error was not recoverable, the build process will terminate and a message will be displayed stating "Build process failed." Otherwise, the messages should be considered informational only.

Memory Allocation Errors. The following message will be displayed in the Messages 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.

Time-Out Errors. The following message will be displayed in the Messages window if it takes too long to evaluate a strategy:

```
 Strategy evaluation took too long (timed out), possibly from too many
trades, Monte Carlo samples, or bars.
```

This means it took longer than the so-called time-out interval to evaluate one of the strategies being built or created during the significance test. The time-out interval is set under **Program Options** (View menu) and defaults to 20 seconds. If it takes longer than this to evaluate a strategy, there may be a problem with the strategy or another issue that is causing the long evaluation time. Common reasons include too many trades (e.g., 20,000 or more), too many bars (e.g., half a million or more), and/or the use of stress testing.

If the problem is too many trades, a simple solution is to select the option to limit the number of entries per day. Reducing the number of bars of data and/or turning stress testing off may resolve the problem. If this error is encountered during population initialization, the program will try to build another strategy to resolve the problem. If the problem continues beyond the maximum number of retry attempts specified on Program Options, the build process will be stopped. Also see **Build Time** in the Usage Topics chapter for more information.

Strategy Generation Errors. The following message will be displayed in the Messages window if there is a problem evaluating a generated strategy:

```
  Strategy generation failed for population member xxx. Creating new
strategy...
```

where xxx is the population member number. This will often be preceded or followed by the following message:

```
Indicator function failed to evaluate Properly.
```

This message most often appears if the TRIX indicator function was evaluated with a non-positive argument, which is not allowed. A non-positive argument to TRIX can occur if the argument is calculated from some other indicator or combination of indicators. When this happens, the program will discard the strategy and generate a new one, as indicated by the error message. These errors are almost always recoverable. Only if the program is unable to generate a valid replacement strategy after the maximum number of retry attempts will the build process terminate.

Build Results Tables

The Build Results tables, which can be seen in Fig. 1.1 in the Introduction, display summary results for the saved population members. There are separate tabs for the main population ("Build Results") and the top strategies population ("Top Results"). A pull-down menu at the top left allows you to select the results for each segment separately or the results for all segments combined. The labels in the pull-down menu are either "training", "test", "validation", and "combined" if the results are over the same data used in the build or "OOS Pre", "In-Sample", "OOS Post", and "combined" if the results are over a different date range (but for the same symbol) than used in the build. OOS Pre corresponds to the portion of the selected date range, if any, that precedes the date range used in building (training, test, and validation). In-Sample corresponds to the date range used in building, and OOS Post corresponds to any data following the date range used in building. If a different symbol is used for an evaluation than used in the build, then all the results are shown as "OOS Post".

The performance metrics displayed in the table are chosen as discussed in the previous chapter. Click on a row in the results table to display the results for the corresponding strategy in the Performance Report, Build Report, Equity Curve, Trade List, and Strategy Code windows. Builder saves and displays the number of strategies you specify on the GP Settings window up to the population size.

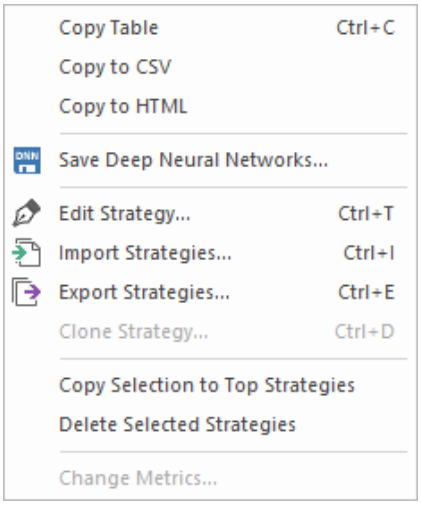
Note that if one segment ends with an open trade that exits in the next segment, then the trade will be included in the metrics for the next segment.

While the build process is ongoing, the program updates the results in the tables after each completed generation. If the build is cancelled before the first generation is completed, the initial population will be displayed. The strategies are listed in order of decreasing fitness on the training segment by default. The table results can be sorted by any column by clicking on the column heading. For example, to sort the results by net profit, click on the Net Profit column label. The results in related tables are sorted the same way. For example, if you sort the training results by drawdown, the test results will appear in the same order.

The Top Results table lists performance results for strategies that meet the filtering criteria specified on the Build Metrics window in the list "Conditions for Selecting Top Strategies". The population of top strategies is updated after each generation is completed for every build within the project file. Top Strategies may come from different builds if multiple builds have been run. Top Strategies are saved in the project file, along with the saved members of the main population. The criteria for selecting the top strategies only apply for trading strategies, not deep neural networks (DNNs), which don't use the strategy metrics. However, you can manually add DNNs to the Top Strategies population from the main population using the context menu, as explained below.

When building a population of trading strategies, the background color of the Build Results tables is determined by the visual theme (e.g., a white background for the Office 2022 Colorful theme). To draw a distinction when building a population of DNNs, each rown in the Build Results tables is colored either green or red, depending on whether the Net Points metric is positive or negative, respectively. In this case, the first metrics listed in the table are the ones shown on the DNN Settings window for the training objectives and conditions, followed by the DNN fitness. Following this are the strategy metrics for the simple strategy that's evaluated as part of the DNN building process, as explained in Deep Neural Networks in the Introduction.

The Build Results tables include a right-click (context) menu, as shown below for the main population table. The menu for the Top Strategies table differs in that (1) the "Copy Selection…" command copies the strategies to the main population, (2) it does not include the "Clone Strategy" option, and (3) there's an additional option to "Delete All Strategies".



If the currently selected row in the table is a DNN, the **Save Deep Neural Networks** command saves it to a file with the extension ".gpdnn". To include a DNN in a strategy during the normal strategy building process, the DNN must first be saved using this command. The DNN can then be selected on the DNN Settings window to be included in strategies, as explained in the Build Settings chapter.

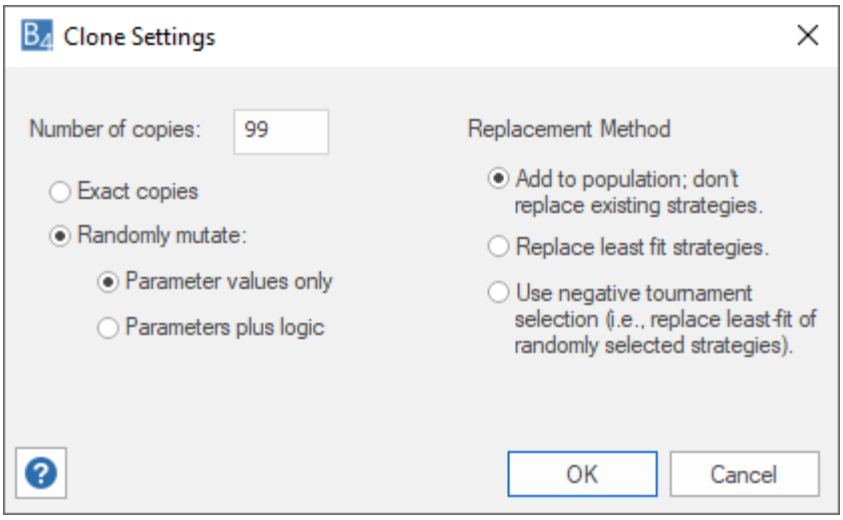
The currently selected strategy can be edited via the included strategy editor program, Adaptrade Editor (see Adaptrade Editor in Build Settings), by selecting the command **Edit Strategy**. This opens the editor program with a temporary copy of the selected strategy. Any changes made in the editor are copied back to the strategy in Builder when the strategy in the editor is saved. Selecting the edit command for other strategies in the Build Results table will open additional Editor windows. Closing an editor window deletes the temporary copy of the strategy.

New strategies can also be created by selecting New from the File menu in Editor. Saving a newly created strategy in the editor writes it to a file with the extension .gpcode. Any such file can then be imported into Builder using the **Import Strategies** command, as explained below. To learn how to use the editor, click the help buttons in the program or press the F1 key while the program is open to bring up the help files. A PDF copy of the user's guide for Editor is available at adaptrade.com.

Strategies can be imported into the table or exported to files using the **Import Strategies** and **Export Strategies** commands. These commands work with files with the extension .gpcode, which store individual strategies -- one strategy per .gpcode file -- and include the strategy specifications and settings needed by Builder (i.e., they include more than just the strategy code in text format). Click the import command to select one or more .gpcode files previously exported or created. The selected strategies will be added to the end of the currently active population, either the main population or Top Strategies. Strategies can be exported from either population using the export command, which will save the strategies to files with the .gpcode extension. You will be prompted to enter a file name "root", such as "MySavedStrategy". If multiple strategies have been selected for exporting, the root will be appended with a number for each saved strategy, such as "MySavedStrategy_1", "MySavedStrategy_2", and so on.

The import and export commands can be used to save noteworthy strategies separately from the project (.gpstrat) file as well as to transfer strategies between project files.

The **Clone Strategy** command adds a specified number of optionally mutated copies of the selected strategy to the main population. Selecting this command opens the Clone Settings window, shown below. You can enter the number of copies to add to the population, whether the copies are exact duplicates of the selected strategy or mutated copies, and the method of adding the copies to the population. If you want to mutate the copies, you can select whether only the strategy inputs are mutated ("Parameter values only") or all elements of the strategy are mutated ("Parameters plus logic"). If the latter is selected, parameter values, entry and exit logic, and strategy inputs will all be subject to mutation using the mutation settings on the GP Settings menu. Position sizing will only be mutated if one of the "Evolve..." options is selected on the Evolve Position Sizing panel of the Position Sizing menu.



Cloned strategies are added to the main population using one of the three methods listed under "Replacement Method". The first method simply adds the copies to the end of the main population shown in Build Results. The second method ("Replace least fit strategies") replaces strategies in the main population based on their training-segment fitness, with the least-fit strategies replaced first, up to the number of copies. If there are more copies than members of the population, the entire population will be replaced. The last option ("Use negative tournament selection") randomly selects a number of strategies from the main population equal to the tournament size setting on the GP Settings menu and replaces the least fit of the selected strategies. This process is repeated for each copy added to the population.

One possible use of the Clone Strategy command is to create a strategy in the editor that you want to explore; for example, a channel breakout strategy. Save the strategy from the editor, then use the import command to add it to an empty population in Builder. Then clone the strategy to fill out the population with mutated copies. Finally, uncheck the **Reset on Build** option on the GP Settings menu prior to building. This will instruct the program to start the build process with the cloned strategies, rather than initializing the population randomly. In this way, the final population will consist of variations of your created strategy (e.g., channel breakout), optimized by the build process. This is a way to use the program to build on your initial idea.

Another application of the clone command is to clone your created strategy to define all members of the population. Then perform an "Evaluate All" and sort by fitness. This provides a sort of random optimization on your initial strategy. Alternatively, if you mutate only the parameter values, you could use the results to see how well the strategy holds up to changes in the input values. This would be essentially the same as performing a stress test on the strategy inputs except that it would show you all the individual strategies, whereas the stress test feature aggregates the results and displays them at the specified confidence level.

To remove a strategy from either of the Build Results tables, click on the entry (or drag to select multiple strategies) then right-click and select **Delete Selected Strategies**. The Top Strategies table can be cleared by right-clicking and selecting **Delete All Strategies**. Neither delete operation will affect the strategies in the other table. Similarly, strategies can be copied from one table to the other using the **Copy Selection to...** commands. For example, this can be used to save an interesting strategy from the main population to Top Strategies before re-building the population in cases where the strategy wasn't picked up by the filtering rules.

The entries in the Top Results table are listed in the order in which they were added to the table. Since strategies can come from different builds, two different strategies may have the same Member number. To see how different strategies in the table compare to one another, click any strategy in the table then select **Evaluate All** from the Evaluation menu. This will re-evaluate each of the strategies in the Top Results table and re-compute their fitness values. You can then sort by fitness by clicking on the Fitness column heading.

The Build Results tables can be copied to a spreadsheet using the **Copy Table** command. Open a blank spreadsheet page and select Paste to insert the table results into the spreadsheet. There is also a right-click menu option to **Change Metrics**, which opens the Change Performance Metrics window, discussed in the previous chapter under Build Metrics.

If stress testing has been applied, the results shown in Build Results will be the Monte Carlo results at the confidence level specified on the Stress Testing window. ***Note:*** *Monte Carlo results of stress testing are calculated metric-by-metric at the specified confidence level and therefore are not directly related to one another. For example, the Monte Carlo net profit is not necessarily the sum of the long Monte Carlo net profit and short Monte Carlo net profit because the long and short net profit are each calculated at 95% confidence. This applies to all metrics.*

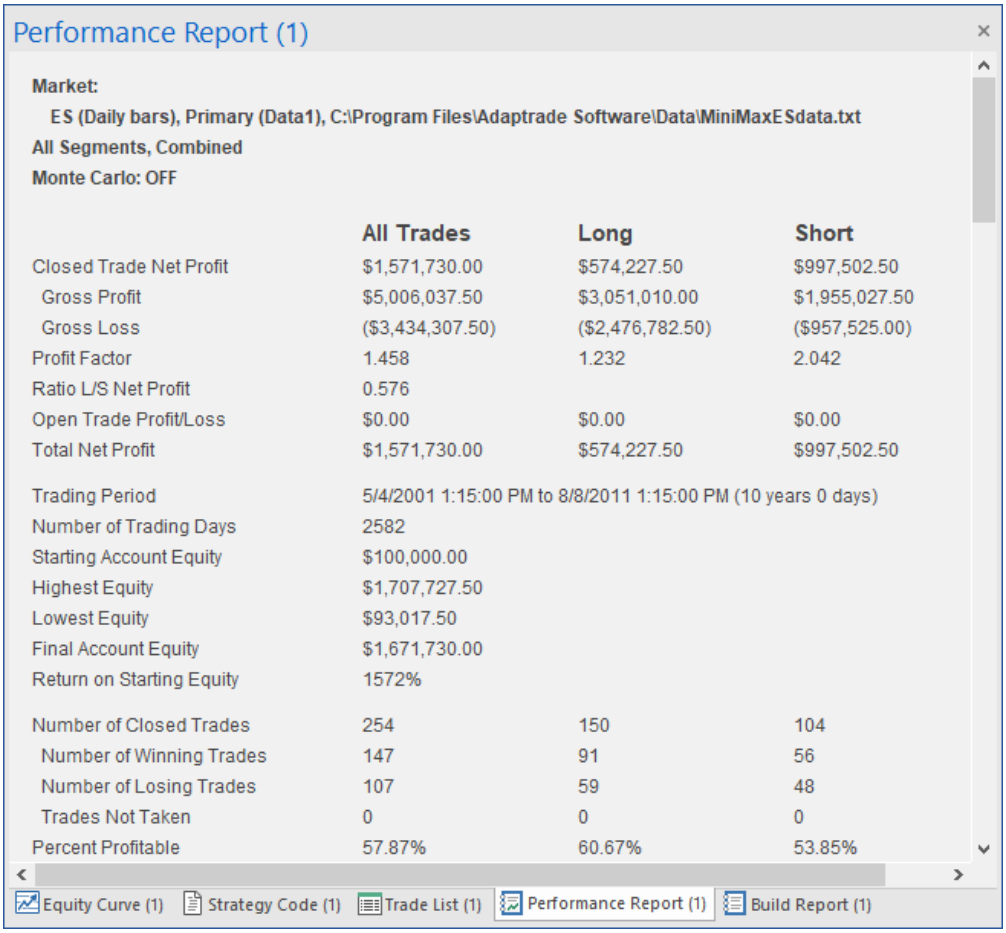
Performance Report Window

The Performance Report window displays a detailed performance report for the strategy selected in Build Results. The performance report can be copied to the clipboard, printed, or saved to a file in several different formats by right-clicking and selecting from the popup (context) menu (**Copy, Save to File, Print, Help**).

An example of part of the performance report is shown in Fig. 6.3. The performance report includes results for all trades together, long trades only, and short trades only. The results are over all data used in the build: training, test, and validation combined. The metrics displayed in the report are described in Appendix: Performance Metrics.

When building a population of deep neural networks (DNNs), the performance results are for the simple strategy that's evaluated as part of the DNN building process, as explained in Deep Neural Networks in the introduction. The DNN-specific metrics are listed at the end of the report.

The market or markets used in evaluating the strategy are shown at the top of the report under Markets. If more than one market was selected in the Market Symbols table, the results will represent the performance of the selected strategy over the combination of markets (i.e., portfolio). To see the results for a different market or markets, change the selections in the Market Symbols table, then select **Evaluate** from the Evaluation menu. The strategy will be evaluated over all markets that have been selected in the Include column of the Market Symbols table.



	All Trades	Long	Short
Closed Trade Net Profit	$1,571,730.00	$574,227.50	$997,502.50
Gross Profit	$5,006,037.50	$3,051,010.00	$1,955,027.50
Gross Loss	($3,434,307.50)	($2,476,782.50)	($957,525.00)
Profit Factor	1.458	1.232	2.042
Ratio L/S Net Profit	0.576		
Open Trade Profit/Loss	$0.00	$0.00	$0.00
Total Net Profit	$1,571,730.00	$574,227.50	$997,502.50
Trading Period	5/4/2001 1:15:00 PM to 8/8/2011 1:15:00 PM (10 years 0 days)		
Number of Trading Days	2582		
Starting Account Equity	$100,000.00		
Highest Equity	$1,707,727.50		
Lowest Equity	$93,017.50		
Final Account Equity	$1,671,730.00		
Return on Starting Equity	1572%		
Number of Closed Trades	254	150	104
Number of Winning Trades	147	91	56
Number of Losing Trades	107	59	48
Trades Not Taken	0	0	0
Percent Profitable	57.87%	60.67%	53.85%

Figure 6.3. Performance report for selected strategy (population member 105).

If stress testing has been applied, the reported results will be the Monte Carlo results at the confidence level entered on the Stress Testing window, and the report will indicate "Monte Carlo: Stress Test Results at x% Confidence", where x is the selected confidence level.

Note: *Monte Carlo results of stress testing are calculated metric-by-metric at the specified confidence level and therefore are not directly related to one another. For example, the Monte Carlo net profit is not necessarily the sum of the long Monte Carlo net profit and short Monte Carlo net profit because the long and short net profit are each calculated at 95% confidence. This applies to all metrics.*

Build Report Window

The Build Report window summarizes the results of the build process for the strategy or deep neural network (DNN) selected in the Build Results table. The build report can be copied to the clipboard, printed, or saved to a file in several different formats by right-clicking and selecting from the popup (context) menu (**Copy, Save to File, Print, Help**).

An example of part of the build report is shown in Fig. 6.4. The build report consists of three sections: Build Summary, Strategy Description or Deep Neural Network (DNN) Description, and Build Settings. The Build Summary includes items such as the build date, time required to build the population, number of generations, etc. Applicable to trading strategies, the Strategy Description lists all elements of the strategy, including indicators, order types, and other strategy features. For DNNs, the Deep Neural Network (DNN) Description is like the Strategy Description except that it also includes the parameters that define the DNN. The Build Settings section records all the settings made by the user that were in place when the build commenced; i.e., all settings that affected the build process.

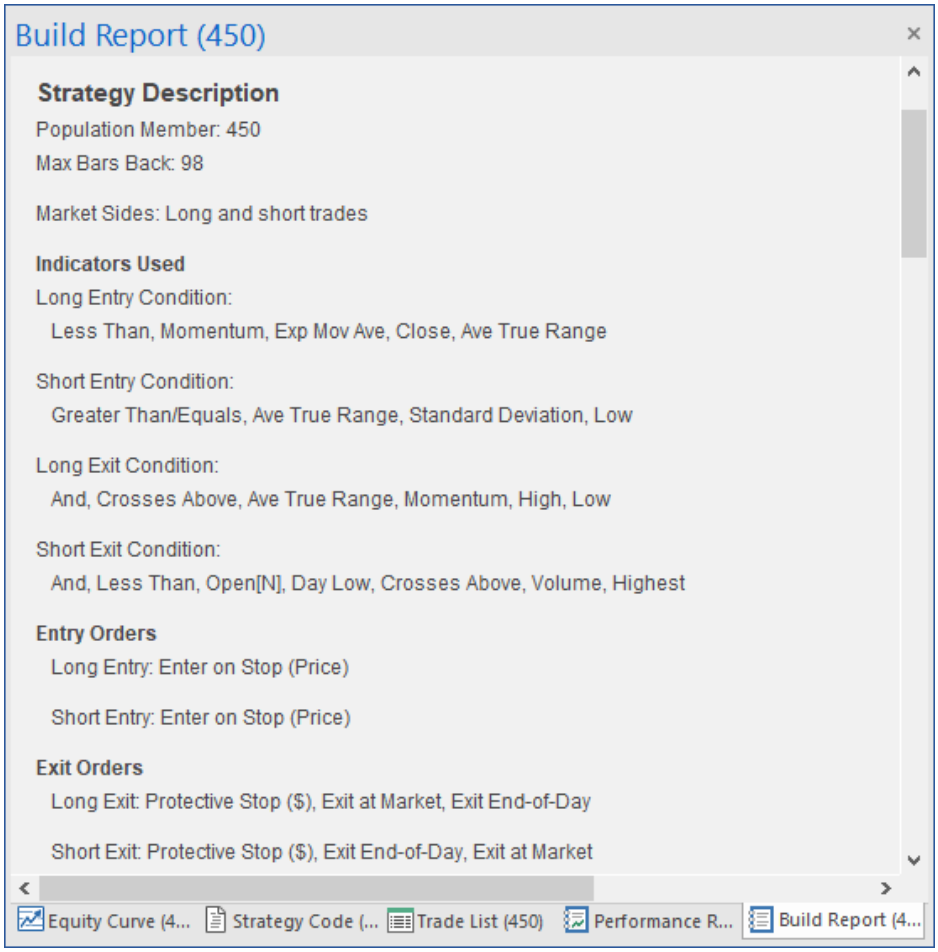


Figure 6.4. Build report for selected strategy (population member 450).

Strategy Code Window

The code for the selected strategy is displayed in the Strategy Code pane. To view the code for a different strategy of the population, select the strategy in the Build Results table. The code in the Strategy Code window can be copied to the clipboard by right-clicking and selecting **Copy Strategy** or by selecting **Strategy Code** from the Clipboard 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. MetaTrader 4 code can be saved directly to a .mq4 file by right-clicking in the code window and selecting "Save MT4 Strategy to File".

Similarly, select "Save NinjaScript Strategy to File" to save the strategy code for NinjaTrader 7/8 directly to a file in the required folder for NinjaTrader 7 or NinjaTrader 8. Once selected, the folder location for NinjaTrader strategies is stored in the registry, so it only needs to be selected once. Builder initially defaults to the folders Documents\NinjaTrader 7\bin\Custom\Strategy\ and Documents\NinjaTrader 8\bin\Custom\Strategies\, which are the customary locations for NinjaTrader 7 and NinjaTrader 8, respectively. Builder creates a default strategy name for each NinjaTrader strategy of the form projectname_member#_#inputs_maxbarsback. For example, for a project with the file name "MyProject.gpstrat", a typical strategy name might be MyProject_3_7_72, which indicates the strategy is from population member 3 with 7 inputs and a MaxBarsBack requirement of 72. When saving the file, a new name can be entered, which will become the name for both the file and the strategy.

Note: The options to save the strategy code directly to a file for either MetaTrader 4 or NinjaTrader are only available if the code is the respective format. Otherwise, the option is grayed out.

Unless saving the strategy code directly to a file, to transfer the code to your trading platform, copy the code from the Builder code window and paste it into the strategy editor of the trading platform. In TradeStation or MultiCharts, open a new strategy window in the EasyLanguage editor, choose a name, then paste the code into the empty strategy window. Finally, compile the EasyLanguage code by pressing F3. In MetaTrader 4, open a new window in the MetaEditor, paste in the code from Builder, and click the compile button. In AmiBroker, in the Charts pane, right-click on the Systems folder and select "New" and then "Formula". Enter a name then right-click on the name and select "Edit". Select "Paste" from the Edit menu to paste the code into the edit window, then click the save icon and, finally, click the AFL icon to verify the code.

If you're pasting the strategy code over the code for an existing strategy, it will be necessary to reset the input values before running the back-test. In MetaTrader 4, this can be done by clicking the Expert properties button on the Tester window and clicking the Reset button on the Inputs tab. In TradeStation, delete the strategy from the chart and re-insert it to reset the input values. In AmiBroker, click the Parameters icon on the Analysis window and click "Reset all".

To test the strategy in TradeStation/MultiCharts, 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; in MultiCharts, Format Signal, Properties) to the **MaxBarsBack** value listed in the results table.

To test the strategy in NinjaTrader 7, first open the strategy in the NinjaTrader editor by selecting Edit NinjaScript from the Tools menu, and click the Compile button. The strategy

should then be ready to back-test via the Strategy Analyzer window. Select the strategy on the Backtest tab of the Strategy Analyzer window and click the button "Run Backtest".

In NinjaTrader 8, provided the strategy has already been saved to the NinjTrader 8 Strategies folder, the strategy should already be compiled. Open a Strategy Analyzer window and select the strategy from the Settings window. Verify the settings and click the Run button.

To test the strategy in MetaTrader 4, select the strategy (Expert Advisor) on the Tester window (Settings tab), and select the symbol and date range. For best results in MetaTrader, select "Every tick" as the Model, then click the Start button to run the back-test.

To test the strategy in AmiBroker, right-click on the strategy in the Systems folder of the Charts pane and select "Analysis". On the Analysis window, reset the parameter values by clicking on the Parameters icon, and check the settings by clicking on the Settings icon. Finally, click the Backtest icon to run the test.

Equity Curve Window

The equity curve for the strategy selected in the Build Results 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 Build Results 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 Evaluation Symbols window, then select **Evaluate** from the Evaluation menu to re-evaluate the strategy. Similarly, the selected strategy can be evaluated over a different market or markets by selecting the market(s) in the Market Symbols table and re-evaluating the strategy.

Vertical lines are drawn on the chart to divide the different segments. If the equity curve is plotting the build results, the label "Training | Test | Validation" is shown in the lower left-hand corner. In this case, the line between the training and test segments is gray, and the line between the test and validation segments is drawn in green if the validation results are profitable or red if they’re not.

If you change the date range on the Evaluation Symbols window after building and re-evaluate the strategy, Builder will display the label "OOS Pre | In-Sample | OOS Post". OOS Pre corresponds to the portion of the selected date range, if any, that precedes the date range used in building (training, test, and validation). In-Sample corresponds to the date range used in building, and OOS Post corresponds to any data following the date range used in building. The OOS Pre and OOS Post segments of the curve use the same green/red color coding as validation data, and the In-Sample segment is colored blue.

The equity curve pane can display either the closed-trade equity curve or the detailed equity curve.

Closed-Trade Equity Curve

The closed-trade equity curve displays the equity at the close of each trade versus trade number. It does not include contributions from any external equity curve symbols that are part of the evaluation and does not display equity values between trade entries and exits.

When multiple primary markets are selected in the Market Symbols table, a separate curve is plotted for each individual market, based on trades from that market. In addition, a portfolio equity curve (i.e., the composite equity curve for all primary price symbols) is plotted with a slightly thicker line, using the same coloring scheme discussed above. A color-coded key

consisting of the symbol names for all primary price symbols is drawn in the lower right part of the plot to help identify which curve corresponds to which market.

When stress testing is applied, a separate equity curve is shown for each stress test, and each curve is drawn in a different color. The equity curve for the original data is plotted with a slightly thicker line. When stress testing is applied to a portfolio, each equity curve line represents the portfolio equity curve for one stress test.

Detailed Equity Curve

The detailed equity curve consists of the bar-by-bar equity values plotted by date; each point on the curve represents the equity value at the close of a bar. This display includes any external equity curves that are part of the evaluation. Regardless of the other settings and options in effect, only one curve is plotted for the detailed equity curve. The plotted curve represents the composite of all symbols, including equity curve symbols, that are included in the evaluation. If stress testing is applied, the curve corresponds to the original (unmodified) values.

The project file does not store the detailed equity curve values for a strategy unless the detailed equity curve is displayed for that strategy when the project file is saved. Otherwise, if the detailed equity curve option is selected, the strategy will be re-evaluated to generate the bar-by-bar equity values before the curve is displayed.

Note: Because displaying the detailed equity curve usually requires a re-evaluation, the symbols on the Evaluation Symbols window should match those on the Build Symbols window to get results that correspond to the build results. If the symbols don't match or no symbols have been selected on the Evaluation Symbols window, an error message will be displayed.

When one or more equity curve symbols are included in the build, the detailed equity curve will be displayed by default. Otherwise, the equity curve pane will default to the type of equity curve -- either closed-trade or detailed -- that was displayed for the selected strategy prior to building.

To switch the display from the closed-trade to the detailed equity curve or vice-versa, right-click on the plot area and select **Plot Detailed Equity Curve** from the popup (context) menu. This will toggle the display from one plot type to the other.

The equity curve can be copied to the clipboard by right-clicking and selecting **Copy Equity Curve** or by selecting **Equity Curve** from the Clipboard menu. Selecting the right-click option **Save to Text File** will save the equity curve in a .csv file that can be added to the symbol library as an equity curve symbol.

Trade List Table

The Trade List table displays the trade-by-trade results for the selected strategy or deep neural network (DNN). To view the trade list for a different member of the population, select the corresponding row in the Build Results table. If the selection is a DNN, the trade results are for the simple strategy used to evaluate the DNN, as discussed in Deep Neural Networks in the Introduction.

The trade list table includes the following fields: symbol, entry date, signal price, entry price, entry signal name, exit date, signal price, exit price, exit signal name, stop price, direction, quantity, profit/loss, costs, net profit/loss, risk and equity.

Entries and exits have both a signal price and the price at which the simulated trade was filled (i.e., Entry Price and Exit Price). The signal price is the price calculated by the strategy and will often be different than the fill price because of where the next bar opened. For entries using stop or limit orders, the signal prices are given by the code variables EntPrL and EntPrS for long and short trades, respectively. For entries at market, the signal price will be the close of the bar on which the order was generated. For exits, the signal price may be a stop price (code variables LStop, SStop), target price (code variables TargPrL, TargPrS), or market price. For exits at market, the signal price will be the close of the bar on which the order was generated. The entry and exit signal names are as shown below in the table.

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”. Quantity is the number of shares or contracts that were traded. The profit/loss field is the closed trade profit or loss before costs, which are the total costs for the position. The net profit/loss is the profit/loss amount after costs. The risk is the potential loss per share or contract based on the protective stop, if any, and includes costs. The equity field displays the cumulative closed trade equity, as shown in the closed-trade equity curve plot. Note that a trade that enters during one segment but exits in the next segment will be listed in the next section of the list. For any trade that did not exit before the end of the trading period, the word "open" will be displayed in the Exit Price column. Open trades are not included in the calculations shown in the Performance Report, except where noted.

Order Type	**Signal Name in Trade List Table**
Enter at Market	Market
Enter on Stop ($)	Fixed Stop
Enter on Stop (%)	Percent Stop
Enter on Stop (Price)	Price Stop
Enter at Limit ($)	Fixed Limit
Enter at Limit (%)	Percent Limit
Enter at Limit (Price)	Price Limit
Exit at Target ($)	Fixed Target
Exit at Target (%)	Percent Target
Exit at Target (Price)	Price Target
Trailing Stop ($ Floor)	Trail $
Trailing Stop (ATR Floor)	Trail ATR
Protective Stop ($)	Fixed Stop
Protective Stop (%)	Percent Stop
Protective Stop (Price)	Price Stop
Exit After N Bars	N Bars
Exit After N Bars Profit	Bars Profit
Exit After N Bars Loss	Bars Loss
Exit After Time	Time
Exit at Market	Market
Exit End-of-Day	End-of-Day
Exit End-of-Week	End-of-Week

Training results in the trade list table are displayed with profitable trades written in green text and unprofitable trades in red text against a background color that depends on the chosen theme. The test trades are displayed with a white text color and the background color set to green or red for profitable or unprofitable trades, respectively. The validation trades are written using the same background coloring as for the test trades but with black text.

The trade list table can be copied to a spreadsheet by right-clicking on the table and selecting **Copy** or by selecting **Trade List** from the Clipboard menu. 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.

To save the trades directly to a MSA (extension .msa) file, right-click and select **Save to MSA file**. The program will prompt you to enter a name for the MSA file as well as a name for the text file to which the trades will be saved. The resulting .msa file can be opened directly by MSA version 3.3 or higher and will include all current Builder settings that are applicable to MSA. If more than one market was used in the build process, the point value and trading costs stored in the MSA file will be the ones for the currently selected market.

Chapter 7
Usage Topics

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 an explanation of stress testing and Monte Carlo analysis, 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, common questions, and various tips and hints for strategy building.

Stress Testing and Monte Carlo Analysis

Testing a trading strategy for robustness is often referred to as sensitivity analysis or **stress testing**. The basic idea is to see what happens when small changes are made to the strategy inputs, price data, or other elements of the strategy or the trading environment. A robust strategy exhibits a proportional and relatively muted reaction to such changes, whereas a strategy that is not robust will react disproportionally and sometimes fail outright when small changes are made to its inputs or environment.

Why is This Important?

Strategy robustness is important because the markets never stay the same. Consider the strategy inputs, for example. Inputs such as the look-back length for a moving average might be optimal over the back-test period, but going forward, different values might be optimal. How will the strategy perform when the inputs are no longer optimal? One way to answer that is to see how the results change when the input values are changed.

Robustness is related to strategy over-fitting. The strategy should not have been fit so tightly to the market during the development process that it can't withstand any changes to the market. To test for over-fitting, the market can be changed, the strategy can be changed, or both can be changed. A strategy that does not stand up well to relatively small changes is not robust and is likely to be over-fit. Such a strategy should not be expected to do well in the future.

Types of Stress Testing

There are many ways that a strategy can be stress tested. Changes can be made to the strategy itself or to the price data on which it's back-tested. The trading costs, such as the amount of slippage, can be changed or the position sizing can be changed. In principle, anything that affects the strategy back-testing results can be varied. In Builder, the following three types of stress testing are available (see Stress Testing the Build Settings chapter):

1. Changing the strategy inputs.
2. Changing individual prices.
3. Changing the starting bar.

The rationale for changing the strategy inputs was discussed above. Two settings are available to randomly change the input values. One is the probability of changing an input. For example, if the probability is 10%, that means there's a 10% chance that an input will be changed. The second setting is the maximum percentage change that will be applied to an

input that is being changed. The actual amount of the change is randomly chosen between -Max and +Max, where Max is the maximum percentage change and might be on the order of 5% or 10%. This percentage is applied to the range of values for each input. For example, if the look-back length for an indicator is selected from the range of values from 1 to 101, then the range would be 100. If the Max value is 10, the randomly chosen change percentage might be, say, -3%, which means a value of -3 (i.e., -3% of 100) would then be added to the original input value to obtain the changed input value.

One way that a strategy can be over-fit, and therefore not robust, is if it's fit too closely to specific prices in the back-test. For example, if the strategy enters long on a stop and several large, profitable trades enter at the high price of the day, that should raise a red flag. What would the results look like if the high had been one tick lower on those days? If such a small change would ruin the results, the strategy is clearly not robust. A stress-testing technique to detect that kind of over-fitting is to make random changes to individual prices and evaluate the results.

Two settings are available to randomly change the price data. One is the probability of changing a price. For example, if the probability is 50%, that means there's a 50% chance that a price -- open, high, low, close of each bar -- will be changed. The second setting is the maximum percentage change that will be applied to a price that is being changed. As with the input values, the actual amount of the change is randomly chosen between -Max and +Max, where Max is the maximum percentage price change. The value of Max is taken as a percentage of the average true range over the past 100 bars. For example, if the average true range is 10 points and the maximum percentage change is 20%, then the change amount is a randomly chosen number between -2 and +2 points. Let's say the actual number is -1.25 points, and the closing price is 1250.50. The modified close would then be 1249.25. Finally, it's possible that changing a price will invalidate the normal price ordering, such as reducing the open so that it's below the low. To prevent that, the prices will be adjusted if necessary after making the change to keep the open and close within the high/low range.

The last available stress testing method involves changing the starting bar. It's probably obvious that a good strategy should not fall apart when you start the back-test on a different bar. It might be less obvious how this can happen. Consider a hypothetical strategy that enters long on a moving average crossover. It then holds the trade exactly five bars before exiting at market. Putting aside the suitability of the logic, imagine what the trade history might look like on a price chart. If the moving average entry condition uses a short-term average crossing above a long-term average, it's entirely possible that in a sustained up-trend, the entry condition could be true for a long period of time; i.e., the short-term average might be higher than the long-term average for many bars in a row.

If the back-test were started during that period, the first trade would enter on the next bar after the starting bar, and each trade would last five bars, followed immediately by the next entry, and so on. Now consider what would happen if the starting bar were changed. If the starting bar were one bar later, for example, the whole series of trades would be shifted one bar to the right. It's entirely possible that some of those series of five-bar trades would be much more profitable than others, depending on how the trades aligned with any underlying five-bar trend cycle that existed. So, depending on the starting bar, the strategy might be highly profitable or unprofitable because of where the trades started and ended. It might not be obvious during development that the strategy logic had this type of dependency on the starting bar, particularly for more complex types of logic.

To test for the effect of the starting bar, the bar on which the strategy back-test is started can be varied by a random number chosen between 1 and N. For example, if N is chosen to be

300, the starting bar will be varied by adding a randomly chosen number between 1 and 300 to the original starting bar number.

Monte Carlo Analysis

Varying the inputs, prices, and/or the starting bar by a random amount only provides one alternative to compare against the original results. To get a more complete picture of how robust a strategy is, the process can be repeated many times to create a distribution of results. Varying the input variables randomly over many samples to generate a statistical distribution of results for the function that depends on those inputs is called Monte Carlo analysis.

In this case, the function is the trading strategy, and the function inputs are the strategy inputs, market prices, and/or the starting bar. By repeating the stress test many times, multiple sets of trading results are generated. A typical equity curve plot resulting from this process is shown below in which each line in the plot represents the equity curve from one sample of the stress test. The thicker green line in the figure represents the equity curve corresponding to the original results.

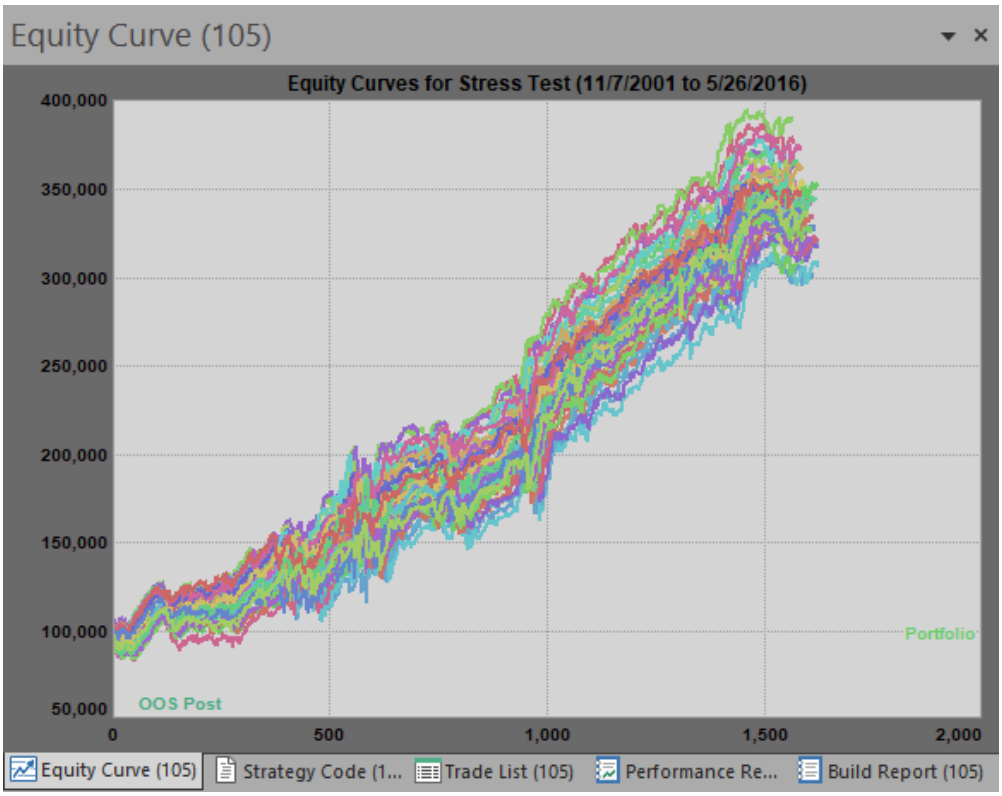


Figure 7.1. Stress test results based on Monte Carlo analysis.

In a different example, suppose that the number of Monte Carlo samples or iterations was set to 20, resulting in a total of 20 sets of results. A typical set of net profit results corresponding to each equity curve might be the following:

147855.00
133286.00
87771.00
92707.00
132149.00
88384.00

126019.00
96581.00
105466.00
102946.00
86753.00
96127.00
116611.00
68459.00
109427.00
96242.00
111020.00
50201.00
130076.00
104181.00

The highest value is $147,855 while the lowest value is $50,201. The Monte Carlo analysis can be used to estimate the likely net profit with a specified degree of confidence given the variety of results. A confidence level of 95% is typical, which means there would be a 5% chance of the net profit being lower than the estimated value. To obtain the value of net profit at 95% confidence, the list above is sorted from highest to lowest, and the value 95% of the way down the list is selected. Since there are 20 items in the list, the 19th item in the sorted list is selected, which would be a net profit of $68,459; i.e., the second lowest value in the list.

This result can be interpreted as follows: if the randomization of the price data is representative of the kind of random differences expected in the market, then 95% of the time, the net profit can be expected to be at least $68,459.

Builder applies the same approach to all performance metrics tracked by the program. If the metric is one where a lower value is better, such as maximum drawdown, the value is selected 95% of the way up the list.

In summary, stress tests measure how robust a trading strategy is, which is an indication of whether the strategy is over-fit. The stress test results are analyzed using Monte Carlo analysis, which makes it possible to quantify the results and provides an estimate of performance that is generally more conservative (i.e., less favorable) than the back-test results based on the original data.

It's also possible to apply stress testing in Builder as part of the strategy development process. Instead of using the performance obtained from back-testing the strategies on the original data, the Monte Carlo results at 95% confidence from the stress tests can be used when evolving the strategies. The top strategies in the population will be the ones with the best Monte Carlo results, which will tend to drive the population towards robust strategies. The downside is that if each Monte Carlo analysis is based on N samples or iterations, the build process will take N times as long using this approach.

Along with out-of-sample testing and other methods discussed in this guide, stress testing provides another tool to help identify robust trading strategies and avoid over-fitting. If applied as part of the strategy building and/or evaluation process, stress testing may help weed out strategies that are overly sensitive to changes in the trading environment, which could help avoid losses and increase the likelihood of success.

Ideal Trades

Ideal trades are ones that can be identified in hindsight and are always "profitable"; the entries and exits are chosen specifically to produce the desired trade result. In general, ideal trades can be used to estimate the potential profit in the market, such as with Pardo's "perfect profit" (Pardo, R. *Design, Testing, and Optimization of Trading Systems*, John Wiley & Sons, Inc., New York, 1992), or they can be used as targets for strategy building.

The ideal trades in Builder are specifically intended to be used as targets for strategy building and are generated automatically by the program based on simple specifications. To use the ideal trades for strategy building, the trades generated by each strategy are compared to the ideal trades to compute the Ideal Matches metric for measuring the degree of correspondence between the two sets of trades. By building strategies that maximize the Ideal Matches metric, the generated strategies will reproduce as many of the ideal trades as possible. This simplifies the specification of build metrics while basing the build process on the market-specific characteristics implicitly defined by the ideal trades.

Defining Ideal Trades for Matching

While there may be several different ways to define so-called ideal trades, the intent in Builder is to define trades that would be desirable to duplicate in real trading. Accordingly, the ideal trades are based on a specified maximum risk and a minimum profit. The risk is defined as the largest open loss prior to the exit, which is also known as the maximum adverse excursion (MAE). Both risks and profits can be defined adaptively, in terms of multiples of the average true range, or fixed dollar amounts can be used. Trades are assumed to be entered and exited at the bar's open.

The key to using ideal trades for matching is to define the trades so that they have a range of bars for both entry and exit. If they were defined with only a single entry bar and a single exit bar, the likelihood of getting matches would be substantially lower and the utility of the matching metric would be questionable.

The process of identifying the ideal trades proceeds as follows:

1. Calculate the risk/MAE at bar i. If the MAE is below the specified risk limit, proceed to the next step.
2. Calculate both the profit and risk/MAE on bar j = i + 1. If the profit is less than the minimum and the MAE is less than the maximum, increment j and repeat this step. If the profit is greater than or equal to the minimum and the MAE is less than the maximum, a valid ideal trade has been found, so go to the next step. If the MAE is exceeded before the minimum profit is reached, a valid ideal trade does not exist at bar i, so increment i and return to step 1.
3. Try to extend the range of exit bars for the ideal trade found in the prior step. Increment j and calculate the MAE. If the MAE is not exceeded, add bar j to the range of exit bars for the ideal trade. If the MAE is exceeded, go to the next step.
4. Try to extend the range of entries for the ideal trade found above. At bar k = i - 1, repeat the steps above. If a valid ideal trade is found at bar k, add this bar to the original trade's entry bar range and add the exit bars to the original trade's exit bar range provided the exit bar ranges overlap. Decrement k and repeat until a valid ideal trade cannot be found or the exit bar ranges no longer overlap. Set k = i + 1 and repeat this step, incrementing k instead of decrementing it at each step.
5. Set i to the bar following the end of the entry bar range for the current ideal trade and go back to step 1.

This process applies to either long or short trades. When looking for both, if a long trade is not found at bar i, the process is repeated for a short trade. By following this process, a set of ideal trades is generated that includes a range of bars for both entry and exit. For each bar in the entry bar range, there is at least one bar in the exit bar range that satisfies both the risk and profit requirements, although there may be many such exit bars. In addition, there may be exit bars where the profit requirement is met that are separated by exit bars that don't meet the profit requirement, provided those bars don't exceed the MAE. This allows the ideal trade to match a strategy-generated trade where the exit occurs later than at the first opportunity, provided the MAE is not exceeded. Also, by searching for the next ideal trade on the bar following the last bar in the entry bar range for the current ideal trade, every bar is examined as a possible entry bar for an ideal trade. This implies that if a strategy-generated trade meets the profit and risk requirements, there is an ideal trade that matches it.

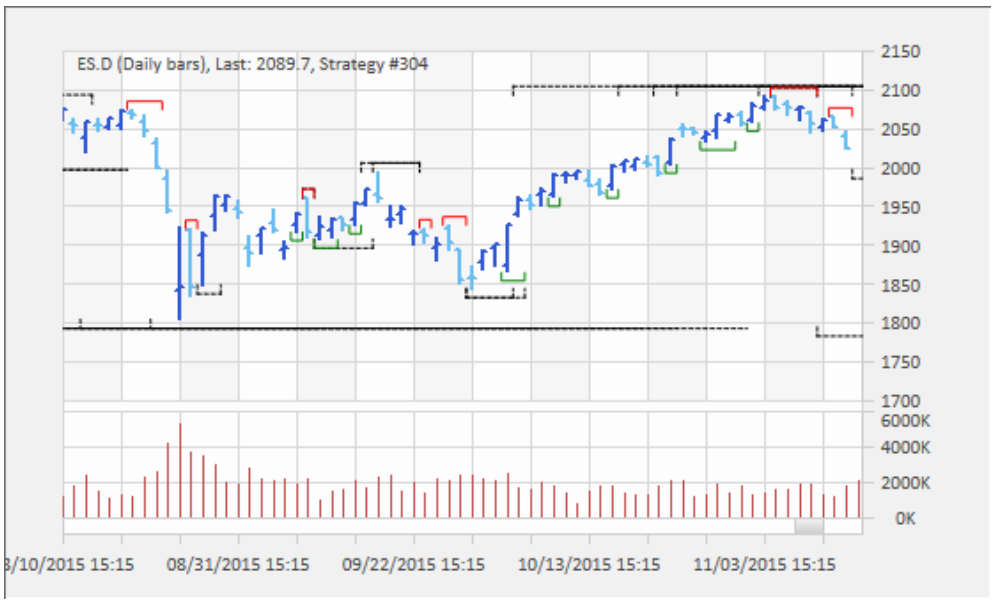


Figure 7.2 Ideal trade markers on a price chart. Ideal trades were defined as a minimum profit of $1500 with a maximum risk of $500. Up and down-pointing solid, colored markers represent long and short entry bar ranges, respectively, while down and up-pointing dotted-line markers represent long and short exit bar ranges, respectively.

Examples of ideal trades are shown in Fig. 7.2 for daily bars of the E-mini S&P 500 futures. The green horizontal bars with up-turned ends positioned below price bars represent the entry locations of long ideal trades, and the red bars with down-turned ends positioned above price bars represent the entry locations of short ideal trades. The black dotted lines represent the exit locations; long exits are positioned above the price bars and are drawn with down-turned ends, and short exits are positioned below the price bars with up-turned ends. Black dotted exit lines drawn without a right-end mark indicate that the exit bars for the ideal trade continue past the end of the line. This generally happens due to a sustained trend. Also notice that whereas entry ranges never overlap, an entry range can overlap with the exit range of one or more other ideal trades, and different exit ranges can overlap. These characteristics enable maximum flexibility in matching the ideal trades to strategy-generated trades.

The Matching Process

Once the ideal trades have been identified, the matching process is straight-forward. A strategy trade matches an ideal trade if the profit of the strategy trade is at least as great as the specified minimum profit of the ideal trade, the MAE of the strategy trade is no greater than the maximum MAE of the ideal trade, the entry bar of the strategy trade lies within the range of entry bars for the ideal trade, and the exit bar of the strategy trade lies within the range of exit bars for the ideal trade. Examples of strategy trades matching ideal trades are shown in Fig. 7.3.

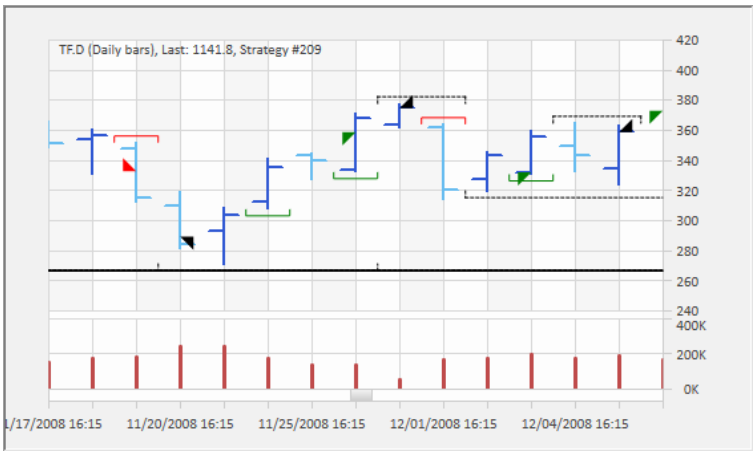


Figure 7.3 Ideal trade matches on daily bars of the E-mini Russell 2000 futures. The minimum profit was set to $1000 with a maximum risk of $500.

To define the "Ideal Matches" metric, the number of matches is combined with the number of strategy trades that don't match an ideal trade. To understand why both matches and non-matches are important, consider two examples. In the first, suppose there are 500 ideal trades and only 10 strategy trades. If all 10 strategy trades match an ideal trade, considering only the strategy matches would imply 100% perfect matching, whereas only 2% of the 500 ideal trades have been matched. In the second example, suppose there are 100 ideal trades but 1000 strategy trades. If all 100 ideal trades are matched by a strategy trade, considering only the ideal trade matches would give the impression of perfect matching, whereas only 10% of the 1000 strategy trades have been matched. Perfect matching can be defined as having the number of matches equal to the number of ideal trades with no non-matching strategy trades. To represent this, the following definition for the "Ideal Matches" metric is used in Builder:

$$\text{Ideal Matches} = 100 * NM/[NI * (a * NN^2 + 1)]$$

where NM is the number of matches, NI is the number of ideal trades, NN is the number of non-matching strategy trades, "a" is a constant, and "^" represents exponentiation (i.e., NN is squared). This equation generates a metric that lies within the range 0 (no matches) to 100 (all ideal and strategy trades match). The term (a * NN^2 + 1) is designed to keep the denominator relatively small for small values so that a small number of non-matching trades doesn't penalize the metric too much. For example, with a = 0.0001 and NN = 100, (a * NN^2 + 1) has the value 2. If, on the other hand, I had used the simpler term (NN + 1), 100 non-matching trades would result in division by 101. Also, the exponent means that for increasing values of NN, the penalty is progressively larger, so a large number of non-matching trades results in a very small Ideal Matches value.

Building with Ideal Matches

There are several different ways to use the Ideal Matches metric for building strategies in Adaptrade Builder.

Ideal Matches as the Sole Metric. On daily bars of the E-mini S&P 500 futures from 9/4/2012 to 3/6/2020 using a minimum profit of $1500 with a maximum risk of $750, there are 207 ideal trades in total. In an example build, this corresponded to 115, 37, and 55 ideal trades on the training, test, and validation segments, respectively. As an example of using the Ideal Matches metric by itself, a single build objective was chosen as the Ideal Matches metric with no build conditions. This tended to generate strategies that had a similar number of trades to the number of ideal trades. After a typical build over 100 generations, the number of strategy trades in the top strategy was 90, 29, and 49, respectively, over the three segments, or 168 trades over the entire date range. The Ideal Matches metric on the training segment of the top five strategies ranged over several different builds from the high single digits to about 25%.

Notably, the number of trades in strategies generated using this approach was relatively uniform within the population and between populations from different builds. This was not the case when building on the same data with the less market-specific objective of maximizing net profit. In that case, it was common to see some top strategies with only a handful of trades in the training segment while others had almost a thousand.

Subjectively, the best strategies were not the ones with the highest value of the Ideal Matches metric. This may be because the ideal trades chosen to illustrate the concept were fixed in size, rather than being adaptive to the market's volatility. As a result, there may have been other trades that didn't fit the definition of "ideal trade" that would have worked better in some cases. Forcing the strategies to match more ideal trades may have prevented it from generating other, more favorable trades.

Ideal Matches with Build Conditions. Another approach is to try to improve upon using the Ideal Matches metric by itself by adding build conditions. In the example above, when conditions were added for the profit factor, percentage of wins, trade significance (i.e., statistical significance of the average trade), and correlation coefficient of the equity curve, this tended to yield better results with more of the performance metrics in a desirable range, as intended by the build conditions. At the same time, the Ideal Matches metric values did not seem to decline for the top strategies, and the number of trades and average trade size still appeared to be controlled by the goal of matching the ideal trades.

Ideal Matches with Build Conditions and Maximizing Net Profit. In the examples above, the Ideal Matches metric had a value less than about 25% in most builds, implying that most trades were not ideal trades. This was true even for the profitable trades. This suggests the idea of adding Net Profit as a build objective (along with Ideal Matches) to more directly influence the trades that don't match an ideal trade. In the example above, setting the relative weight of the Ideal Matches metric to about 0.2 or 0.3 of the weight for Net Profit still allowed the Ideal Matches metric to influence the results while producing better results overall.

The primary effect and benefit of using the Ideal Matches metric is that it provides a way to specify market-specific build conditions related to trade metrics, such as the average trade size, number of trades, average bars in trades, and so on, without having to determine appropriate values for the individual metrics. By specifying just two values, the minimum profit and maximum risk, which can be verified visually on the price chart, the resulting ideal trades provide targets for strategy building. Even though only a fraction of the ideal trades may be matched by any strategy, the goal of matching them guides the build process to strategies with metric values that generally reflect the ideal trades.

Compared to other methods that use ideal trades in strategy building, such as using the equity curve or the metrics that correspond to ideal trades, the method available in Builder is more direct in that it's based on matching the trades themselves. Accordingly, it seems to steer the population of strategies towards ones with metrics representative of the ideal trades even in cases with low match percentages.

The best way to use the Ideal Matches metric may be to combine it with a nonspecific build objective, such as maximizing net profit, and several general build requirements, such as ones based on profit factor, percentage of winning trades, trade significance, correlation coefficient, etc. Conditions or requirements based on these types of metrics are relatively easy to define compared to metrics that are more market-specific, such as the average trade size, number of winners, and so on. For example, if a trader likes to see a profit factor of at least 1.7 or at least 60% winners, these can be used as conditions. Additional information on nonspecific metrics is available in the article Strategy Quality Metrics Improve Out-of-Sample Results, available at adaptrade.com under the Newsletter link.

Additional information on ideal trades and their settings in Builder is available in the Price Charts chapter, Creating and Using Price Charts.

Out-of-Sample Performance

Genetic Programming (GP) is a type of optimization, similar to but more generalized and complex than the simple strategy input optimization that most systematic traders are probably familiar with. When a strategy is developed based on GP, the optimization process fits the strategy to the data used to develop it. The question is whether the strategy will perform well on data not used in the development, such as in real-time trading. Data not used in the development is referred to as out-of-sample data. Clearly, we want good performance on out-of-sample data. This section discusses the factors that affect out-of-sample performance and what can be done to improve it.

Poor out-of-sample performance can be due to several factors. In a program like Builder, which uses a sophisticated search process to find the best strategies, the results are biased by the fact that the search process selects the best results out of all strategies generated. This introduces the so-called *data mining bias*. Builder includes a unique test of statistical significance to detect this and determine if the strategies are in fact unduly biased in this way. If they are, a strategy that appears to generate good results may be benefitting mostly from good luck (technically, from random, sampling error), which suggests that it won't be likely to hold up well out-of-sample. This test was discussed in Evaluation Options in the Build Settings chapter.

Another factor that can degrade out-of-sample performance is *over-fitting*, which refers in this case to the problem of optimizing a strategy so that it fits the training data well but doesn't generalize well to the out-of-sample segment. Any part of the data that can be traded profitably can be considered the *signal*, while everything else can be considered *noise*. Over-fitting is fitting the noise, rather than the signal.

There are two general approaches to dealing with over-fitting. One approach is to detect over-fitting after-the-fact and discard any strategies that are over-fit. The other is to take steps to avoid developing over-fit strategies in the first place. The previous section discussed using stress testing to avoid over-fitting by building based on the Monte Carlo results of stress testing. It was also discussed how stress testing can be used after-the-fact to assess strategies for over-fitting.

Another approach to avoid over-fitting is to use three data segments: training, test, and validation. In Builder, the first two segments are utilized by the Build Failure Rules and Build Termination Options on the GP Settings window of the Build menu. The training segment comprises the data used to build the strategies. The second segment, known as the test segment, is used to monitor the build process and minimize over-fitting. The validation segment is not used to build the strategies. Validation data are reserved for out-of-sample testing following strategy building. The results on the validation segment provide an unbiased estimate of strategy performance because the validation data were not used in building the strategies.

The Build Failure Rules and Build Termination Options help avoid over-fitting by allowing you to define rules that stop or reset the build process when the results on the test segment start to decline. If the performance on the training segment continues to improve as the build process progresses, but the results on the test segment start to level off or decline, it suggests that the strategies are being over-fit. In other words, they're fitting the data used to develop them better but are not generalizing well to other (i.e., test) data. These rules can be used to detect this situation and either reset the build or terminate it.

Another way to avoid over-fitting is to guide the build process towards better quality strategies by targeting metrics related to over-fitting and general strategy quality. One factor is the so-called *number of degrees-of-freedom*. The number of degrees-of-freedom, which is equal to the number of trades minus the number of rules and conditions of the strategy, is related to 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 adding requirements for the number of trades and/or the strategy complexity. All other things being equal, adding a build condition for the number of trades will result in strategies with more trades and therefore more degrees-of-freedom. Likewise, adding a build condition for the complexity metric will result in strategies with fewer inputs, which will also increase the number of degrees-of-freedom.

The degrees-of-freedom can also be targeted indirectly via the Trade Significance metric (not to be confused with the statistical significance test discussed above). The Trade Significance (or simply Significance) metric is based on the Student's *t* test applied to the average trade. The *t* test is based on the number of degrees-of-freedom but is a more complete measure of strategy quality than the number of degrees-of-freedom alone. One way, then, to reduce the likelihood of over-fitting is to use the significance metric to generate strategies that have a high Significance, such as greater than 95%.

Another measure of strategy quality is consistent results across different periods. 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 regions 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 adding this metric as a condition and requiring a high value, such as 0.95.

Another factor affecting strategy quality is the variety of market conditions in the training segment. It's usually 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 training 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 this doesn't necessarily relate to over-fitting -- you could still fit the noise in each of the different market conditions -- a strategy that is trained on a wide variety of market conditions and is not over-fit has a better chance of generalizing well to new data than a strategy that was trained on only one set of market conditions.

Unfortunately, there will be cases where even with good performance metrics, a wide variety of market conditions in the training segment, and no evidence of over-fitting, the out-of-sample performance will be poor. This can happen for several reasons. First, the strategy might be *under-fit*. While over-fitting is probably more common, it's possible that the strategy might not be complex enough to capture the underlying market dynamics. This would usually show up as poor performance in the training segment, which would result in rejected strategies. However, there may be cases where the market dynamics extend into the out-of-sample segment and are not captured by the training data. An analogy would be fitting a straight line to a scatter plot that appears to be linear based on the first half of the data but can be seen to be curving upward when all the data are seen together. 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 re-optimize the strategy inputs without rebuilding the strategy, including the most recent data in the optimization. Then test the new strategy inputs out-of-sample by tracking the performance in real-time. In most cases, a strategy that has many trades, a high significance value and good performance on the training segment will continue to perform well for some period of time post-optimization. The next question is how long it will continue to perform before requiring re-optimization or rebuilding. This issue is discussed below 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 and evaluate 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.
- Number of trades. Strategies with many trades take longer to evaluate. One factor that can result in an unusually large number of trades is not specifying adequate trading costs. With zero or small trading costs, numerous very short-term trades with very small or even zero profit/loss values will not negatively impact the fitness. For example, with zero trading costs, a top strategy could have thousands of trades that enter and exit at the same price, resulting in zero profit/loss with no impact on the strategy's fitness. To avoid

this, always specify a reasonable value for the trading costs and/or select the option to use the bid/ask spread to determine fills on the Evaluation menu.

- 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 for a preliminary build to several hours or even days for a large price file. If the build process seems to be taking too long, it can be cancelled and restarted later. 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 (GP Settings, Build menu). 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 on the GP Settings window. To restart the build process exactly where you left off, the number of saved strategies 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.

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 developed strategies prior to real-time trading. If the Build Failure Rules and Build Termination Options are not used, the test segment will be out-of-sample. However, if either the Build Failure Rules or Build Termination Options are used, the test segment cannot be considered out-of-sample. In this case, it's important to utilize the validation segment in Builder for out-of-sample testing. The default settings on the Build Symbols window of 60/20/20 for the training/test/validation percentages are usually reasonable.

To select the best strategy from the final population, consider sorting the results by the fitness on the test segment. This can be done by selecting the test segment from the drop-down menu on the Build Results table, then clicking on the fitness column so that the results are sorted from highest to lowest fitness. Selecting the strategy with the highest fitness on the test segment will minimize the so-called training bias (see, for example, "Is Your Account Size Large Enough for Your Strategy?" under the Newsletter link at www.Adaptrade.com). There can still be selection bias, however. That's why it's necessary to verify the results by checking that they hold up on the validation segment.

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 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 the trading platform.

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 Evaluation Symbols window. After adding the new market, select the strategy to evaluate and select Evaluate from the Evaluation 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, including, of course, building over multiple primary symbols.

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. For example, 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 Parameter Ranges window on the Build menu in Builder. For example, if an input represents a multiple of the average true range for an entry price, you can use the range of values under “Price Difference Multiple, Entries” in the Parameter Ranges window. Likewise, if an input is a look-back length for a price pattern, you can use the range of values under “Lookback, Price Patterns”. 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 your trading platform rather than rebuilding in Builder. If that doesn’t work, a new strategy can always be built in Builder.

Common Questions

Why can't I open Adaptrade Editor from within Builder? When I try, it gives me an error message and the editor doesn't open.
This is due to a file permissions issue. To resolve the problem, try right-clicking on the setup file (e.g., BuilderLicensedSetup_v500.exe for the licensed version or BuilderTrialSetup_v500.exe for the trial) and selecting "Run as administrator". It should not be necessary to uninstall the current installation first. Other methods that can sometimes fix the problem include running Builder by right-clicking the program icon on your desktop and selecting "Run as administrator" or directly changing the file permissions for the files associated with Builder. The files associated with Builder include the files installed in Program Files\Adaptrade Software\Adaptrade Builder x.x.x (where x.x.x is the version number) and in Documents\Adaptrade Builder. Also try: (1) turning off User Account Control in Windows, (2) creating a new Windows user account with administrative privileges, or (3) scanning to make sure you don’t have a virus.

In some cases, the results displayed by Builder don't match the results I get when I run the strategy in my trading platform. What's wrong?
There are several possible explanations. One possibility is that the date ranges of the price series are different between Builder and the trading platform. It's also important to set the "Maximum number of bars study will reference" in TradeStation (Format Strategies, Properties for All) to the MaxBarsBack value listed in the Build Results table. In MultiCharts, the value is entered under Format Signal, Properties ("Maximum number of bars study will reference"). This will ensure that the calculations start on the correct bar. This is not necessary in AmiBroker since the code itself contains the MaxBarsBack setting.

MetaTrader automatically starts the indicator calculations so that the first bar of trading is on the first day you specify. This means you don't need to enter a "max bars back" value. However, it also makes it difficult to align the starting date in Builder with that of MetaTrader. Sometimes, the bar on which the calculations start can make a substantial difference. For example, if trading starts two bars earlier in MetaTrader than in Builder and each trade lasts exactly five bars, each trade could be starting and ending two bars earlier in MetaTrader, which could significantly affect the results.

Trading costs can also be a factor. Note that Builder deducts the trading costs once per trade, whereas TradeStation deducts costs once "per side". For example, if your trading costs are set to $25 in Builder, they should be set to $12.50 in TradeStation.

If you're re-using a strategy name in the strategy editor by pasting the strategy code over the code for an existing strategy, it will be necessary to reset the input values before running the back-test. Otherwise, the trading platform may be using the prior strategy's input values with the new strategy. In TradeStation, delete the strategy from the chart and re-insert it to reset the input values. In MetaTrader 4, reset the input values by clicking the Expert properties button on the Tester window and clicking the Reset button on the Inputs tab. In AmiBroker, click the Parameters icon on the Analysis window and click "Reset all".

MetaTrader 4 applies a minimum price distance to determine if an order can be placed. If a pending order (stop or limit) is too close to the market at the time it's placed, the order will be rejected. This is based on the idea that there won't be sufficient time to place the order before the market moves through the order price. Builder doesn't reject such orders, which can sometimes cause a discrepancy in back-testing between Builder and MetaTrader 4.

Another source of difference in MetaTrader involves the accumulation/distribution indicator. The values of this indicator depend on the bar on which the calculations start. In MetaTrader, while you can specify the starting and ending dates for the strategy, the indicators are evaluated starting at the beginning of the available data. This means that the accumulation/distribution values may be very different in MetaTrader than in Builder.

In TradeStation, another possible source of difference is how Builder and TradeStation use trading volume on intraday data. If your strategy uses volume in an indicator for either an entry or exit condition and your price data is intraday data, TradeStation may be using only the "up volume", rather than the total (up plus down) volume. There is an option on the Data File Format window to "Set volume to sum of up-tick and down-tick volumes." Try changing this option then re-evaluate the strategy using the Evaluate command on the Evaluation menu.

For forex trading, one source of discrepancy is the so-called "roll over credit", which TradeStation calculates on foreign exchange trades. This is currently outside the scope of

Builder. In most cases, these credits are small amounts that do not affect the overall results substantially.

Also keep in mind that Builder does not look inside bars (what TradeStation refers to as "look-inside back-bar testing"). If your strategies enter and exit on the same bar or have multiple exit orders that could be triggered on the same bar, such as a target and a protective stop, this can result in discrepancies. It's probably better to use more data for a given number of trades during a given trading period rather than less. This suggests that going to a smaller bar size with more bars per trade is preferable to having trades enter and exit on the same bar or having the bar size be so large that both stops and targets could be triggered on the same bar. Provided this is done, there will not be any ambiguity about which order is triggered first, and the discrepancy associated with the lack of look-inside bar testing can be avoided.

Even with all the settings correct, there may be differences. Oftentimes, these result from rounding errors and other small differences that are unavoidable. For example, if an oscillator is compared to a fixed value, such as 80.0, using the less-than-or-equal (<=) operator, there may be a bar where the value calculates as exactly 80.0 in Builder but as 80.0000001 in TradeStation. This seemingly insignificant difference can result in a trade that is not taken in TradeStation but is taken in Builder. This could then affect subsequent trades because the presence or absence of a trade can either prevent or allow another trade to be taken. This is particularly true when entries are taken at market, especially when the entry condition is "true". Such strategies rely on the exit logic, so that if one trade is off by a bar, the next trade will be delayed by one bar, and so on. In this way, a small calculation difference can lead to entirely different results. One way to handle this possibility is to stress test strategies after building them. Strategies that do poorly under stress testing may be too "fragile" for real life trading.

Why does Builder not use look-inside back-bar testing?
As noted in the answer to the previous question, looking inside the bar can only make a difference if there are multiple entries and/or exits on the same bar, which usually only occurs if the stops and/or targets are too small for the bar size. Adding the ability to use intrabar data would slow down the build process considerably and complicate the data requirements for the user.

When I tried to run a strategy in TradeStation, I got an error message "Tried to reference more bars than allowed by current Max bars back setting." In MultiCharts, the error is "Tried to reference back more bars than allowed by current MaxBarsBack setting."
The max bars back (MaxBarsBack) setting in TradeStation and MultiCharts refers to the number of bars required to start the calculations. You need to set this to the value listed in Builder in the results table under MaxBarsBack. The same value is also listed in the comment block at the top of the strategy code. In TradeStation, you enter the value under "Maximum number of bars study will reference" in Format Strategies, Properties for All. In MultiCharts, the value is entered under Format Signal, Properties ("Maximum number of bars study will reference"). This will ensure that the calculations start on the correct bar.

I selected the exit end-of-day order type, but my trades don't exit at the end of day in real time trading. What can I do?
First, the exit end-of-day order type is only available in TradeStation and MultiCharts. For other code types, the "Exit after" option on the Strategy Logic window should be used to exit intraday strategies at the end of the day or at a specified time. For TradeStation and MultiCharts, the "Exit end-of-day" order type on the Order Types window in Builder causes the program to include the SetExitOnClose command in strategies. This command is mostly for back-testing purposes. In real time trading, it causes a market order to be generated on the

close of the last bar of the current session. However, by the time the order is sent, the market has already closed, so the order is not filled. A work-around technique that some traders use is to define a custom session that ends a few minutes prior to the actual session close. Then the SetExitOnClose command will have time to exit your position prior to the end of the actual session. You just need to make sure to correctly set the session end time in Builder for the custom session, and, of course, the chart in TradeStation/MultiCharts has to use the same custom session as in Builder in order to avoid discrepancies between the results in Builder and those in TradeStation/MultiCharts.

A simpler approach is to use the "Exit after" option on the Strategy Logic window. Set the time to at least one bar prior to the end of the session. That will trigger an exit prior to the end of the session.

I want my strategies to have smaller losses. How do I get tighter stops in Builder?
If the stops are too large, the easiest solution is to reduce the maximum stop size on the Parameter Ranges window. There are separate parameter ranges for fixed size, percentage, and price-based stops. You can also set a target for the Max Loss metric. For example, if you want no more than 9 points loss on trades in the E-mini S&P, you can set a target for the Max Loss metric of $450 (i.e., 9 points x $50 per point). Alternatively, you can try including the Ave MAE (maximum adverse excursion) in the build goals. The Ave MAE is a measure of the intra-trade drawdown. There is also a Max MAE metric (maximum value of the MAE over all trades).

If the problem is that you're not getting protective stops in your strategies, you can include any selected exit type in all strategies. To include a protective stop, just select the protective stop in the "Include" column on the Order Types window.

Why do my trades exit past the time I set in the "Exit after" option?
The exit time option allows you to set an exit time for your trades. If, for example, you set the time to 3:00 pm, the strategy logic will be modified so that exits take place after this time. However, it's important to keep in mind that the time stamp on bars is typically the time the bar closes. For example, if you're using 30-minute bars, a bar time of 3:30 pm means the bar closes at 3:30. Since it's a 30-minute bar, that means it opens at 3:00 pm. The exit time option causes a trade to exit on the next bar's open if the time of the current bar is greater than or equal to the exit time. So, if you've specified an exit time of 3:00 pm, the trade will exit on the open of the 3:30 pm bar (assuming 30-minute bars). The exit time will be listed as 3:30 pm because that's the time stamp on the bar, but the actual time will be a moment after 3:00 pm, which corresponds to the open of the 3:30 pm bar.

Also keep in mind that a trading strategy can only trade the bars of price data it has. If you're using 60 minute bars that end on even hours (e.g., 2:00 pm, 3:00 pm, 4:00 pm, etc.), and you specify an exit time of 3:15 pm, your trades will not end at 3:15 pm because there's no bar with that time. In this case, the trade would exit on the open of the 5:00 pm bar, which would be a moment after 4:00 pm.

If your data has been exported from MetaTrader 4, the bar time is the time the bar opens. However, the logic works the same way as described above, and the results should be the same for the same exit-time value.

I specified no more than three entries per day but get four. Why?
The option to limit the number of entries per day uses the EntriesToday function in EasyLanguage or the equivalent code for other languages. The strategy code checks to make sure the current value is less than the specified limit before placing the orders. However, it can still place two orders – one long and one short – if both entry conditions apply, either of

which or both may be filled, depending on the market. That means you could get up to two entries if both are filled, which could result in as many as four entries for the day.

Why does Builder sometimes seem to replace my top population member during the build process?
The program never replaces the top strategy in the population. However, after new strategies are added, the top strategy may change, which means the top strategy from a prior generation may no longer be the top strategy in the current generation. The strategy to replace at each step is chosen as the least fit member among a small number of randomly chosen members. The number of such members is given by the so-called tournament size, which can be changed on the GP Settings window. To decrease the likelihood that one of the top strategies might be replaced, you can increase the tournament size. The default size is 2. However, increasing it to a high number is not recommended and may hurt the performance of the program.

Why are most of my final strategies similar?
If many or all of your final strategies are similar, you probably need to reduce the number of generations and/or utilize the rules under Build Termination Options on the GP Settings window. After several generations, the results will often tend to converge, resulting in similar strategies. This can also happen if insufficient variety is in the initial population. To get more variety in the initial population, either increase the population size or make sure that the build sets (indicator and order) have not been overly restricted by removing too many items. It may also be possible to increase diversity in the final set of strategies by increasing the mutation rate and decreasing the crossover rate. Also, consider using two or more subpopulations.

How do you know when a strategy is broken and needs to be rebuilt or re-optimized?
Depending on the market, time frame, and other factors, it's always possible that a trading strategy will stop working at some point and need to be rebuilt or re-optimized. Please see the section Post-Build Testing and Optimization. One approach involves tracking the trailing performance characteristics of the strategy and rebuilding or re-optimizing when the results start to differ significantly from the long-term averages. A simpler approach is to track the equity curve and look for deviations from straight-line performance. If the strategy needs to be rebuilt, it's just as easy with Builder to create a new strategy as it was to develop the original one.

One point to keep in mind: if you try to re-optimize a strategy's parameter values before the strategy starts to underperform, there's no reason to think you'll get different results (assuming it's already optimized) if you re-optimize over the same data. You can either re-optimize over more recent data while dropping the earlier data from the date range or wait until the strategy has been underperforming sufficiently that the optimization process is likely to find better parameter values over the same data than those you previously found.

Does changing the settings in the middle of a build affect the build process?
No. Any changes made to the settings, such as build goals or strategy options, have no effect on an ongoing build. However, if you stop the build then restart it, any changes you've made will be picked up and used in the new build.

When I compile my MT4 strategy, I get warning messages about functions that have been deleted from the file. Is there something wrong?
No. Those messages can safely be ignored. The include file for MetaTrader 4 contains functions for all the different order types any strategy created by Builder might need. No single strategy will use all the different order types, so the MT4 compiler removes the functions for the order types that are not used. The warnings are simply telling you that those

functions have been removed from the compiled strategy code. The functions have not been removed from the include file itself.

Why are some options not available when I select "AmiBroker" as the code type?
The AmiBroker Formula Language (AFL) makes it very easy to write simple strategies, provided they're consistent with the underlying design of AFL. However, more complex strategies can be very difficult, if not impossible, to construct. The degree of complexity required by Builder-generated strategies makes it impractical to allow AFL strategies that include both long and short trades in the same strategy. Of course, you can generate separate populations for long-only and short-only strategies if you plan to trade both sides of the market. Because strategies can only be long-only or short-only in AmiBroker-generated Builder code, the option to "Wait for exit before entering new trades" is true by default and would be inapplicable if unchecked.

When I compiled my strategy, it didn't recognize one of the functions in the Builder-generated code. What happened?
Builder includes several functions and indicators that need to be installed into the trading platforms for Builder-generated strategies to run properly. If these functions are not installed, the strategies generated by the program will not compile or run. These functions must be installed separately for each platform you intend to use, such as TradeStation or NinjaTrader. Please refer to the installation instructions in the user's guide in the Getting Started chapter.

When I try to start the program, it opens then immediately shuts down. How can I fix this?
Most likely, there's something wrong with the project file the program is trying to open when it starts up. The project file may have been corrupted or damaged for some reason. If you know which project file it's trying to open -- the last one open when you closed the program -- you can delete it or rename it. Otherwise, you should delete the registry entry for Builder in the Windows registry, which tells Builder which file to open when it starts up. This can be done using the Windows regedit program. Please refer to the user's guide (Getting Started, Program Layout and Configuration, Deleting the Registry Entry) for details.

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 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 metrics and build again. Nonetheless, you’ll soon find the combination of settings 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.

1. Include complexity as a build objective with a small weight to rank the simpler strategy higher when two strategies have equivalent logic but different numbers of inputs. This will help to keep the entry conditions from getting unnecessarily verbose.
2. 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.
3. If you'll be running the program overnight or while it’s unmonitored for an extended period of time, consider using the "Performance Failure Rule" on the GP Settings window to reset the build process after every N generations so that the program will

keep working if it doesn't find good results initially. For example, you could use the second build failure rule to start over after every N generations if the average net profit on the test segment is less than 10000000. Because such a high net profit will never be reached, it will start over every N generations. Used in conjunction with the Top Strategies conditions, you can set up the program to save good strategies continuously. In this case, also consider using the option to save the project file after each generation (Auto-Save option, Program Options, View menu) to avoid losing your results in the case of a power outage, computer crash, or other unexpected event.

4. 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 use an objective 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.
5. When setting the build metrics, 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 results and add or remove metrics, change the weights, and add or remove conditions 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 without intervention.
6. It's often better to use a small number of build objectives, such as the ones provided by default, and put the requirements as build conditions.
7. Whether or not you use the Significance metric, a Significance value of at least 95% is generally desirable. This implies adequate average trade size, low standard deviation of the trades, and an adequate number of trades.
8. 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.
9. Always check the bar type and session times on the Symbol Properties when adding a new symbol. Builder tries to infer these settings, but they can be changed if necessary.
10. The point value (Symbol Properties) should usually be 1.0 for both stocks and forex. For futures, the point value is dependent on the contract and can be obtained from the exchange's web site or from your broker.
11. The bid/ask spread is particularly useful for forex trading, where the cost of the trade is often determined from the spread. Set the bid/ask spread in price units on the Symbol Properties window.
12. The Costs column on the Build Symbols and Evaluation Symbols windows corresponds to the Cost Type entry on the Symbol Properties window. To change the cost type -- from, say, Amount per Share to Amount per Trade -- double-click the symbol in the table to open the Symbol Properties window.
13. Inputs for custom indicators must be specified and are not varied by the program. To have the program consider different input values, provide multiple columns for the custom indicator, one for each input value.
14. To include other symbols in the logic for your primary symbol, add the other symbol to the Build Symbols window and change the Series column value to something other than "Primary".
15. The position size for forex trading is typically 10000 shares for a mini lot (equivalent to 0.1 lots in MetaTrader 4) or 100000 shares for a full-size lot.
16. To trade in round lots of, say, 100 shares, set the "Round position sizing" option to 100, set the minimum number of shares to 100, and set the maximum number of shares to a multiple of 100.
17. When building a strategy over a new market, it's usually better to select Fixed Size position sizing with a specified size initially, rather than letting the program evolve it.

Additional complexity, such as variable position sizing, can be added in subsequent builds.

18. The End-of-day exit type (Order Types window) should not be used in most strategies for real-time trading because the exit order will not be placed until the market has already closed, making a fill impossible. Use the entry and exit time options on the Strategy Logic window instead.
19. Only one type of entry order can be used in a strategy (and only one entry per trade). If multiple entry order types are checked in the Include column (Order Types window), only the first one with a check mark in the Include column will be used.
20. If more than one type of protective stop order is selected in the Include column (Order Types window), the stop order type will be selected randomly from the included ones. Likewise, if more than one type of target exit order is selected in the Include column (Order Types window), the target exit order type will be selected randomly from the included ones.
21. Including the MAE (maximum adverse excursion) as a build objective or condition can be an effective way to ensure strategies don't have large open losses before exiting.
22. Builder doesn't directly control leverage in position sizing. However, you can set a build condition (i.e., requirement) for the average or maximum leverage.
23. For strategies that trade both long and short, the metrics "Ratio L/S Profit" and "Ratio L/S Trades" can be used as build conditions or objectives to help guide the population towards strategies with a better balance of long and short-side performance.
24. Choosing how much data to build over is a matter of balance. Too little data increases the likelihood that the strategies will be over-fit to the market(s). Too much data makes it more difficult to find strategies that work consistently well over a wide variety of conditions.
25. Many strategies being saved to the Top Strategies windows suggests the conditions for selecting these strategies (Build Metrics) are not restrictive enough. Try adding new conditions or increasing the requirements for selection.
26. Because most forex symbols trade around-the-clock, the End-of-day exit does not usually make sense for forex. Use the entry and exit-time options on Strategy Logic instead.
27. When the fixed-size protective stop ("Protective Stop ($)") is selected for either consideration or inclusion in strategies, always set the range, such as with the Use Defaults button, on Parameter Ranges.
28. To use a fixed-size protective stop ("Protective Stop ($)") of a particular size, set the range on Parameter Ranges so that the minimum and maximum are the same. The same can be done to specify a percentage protective stop ("Protective Stop (%)") of a particular size.
29. To build better intraday strategies, try limiting the number of entries per day using the options on the Strategy Logic window. 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. Similarly, try restricting the time ranges for entry and exit using the options on the Strategy Logic window.
30. To assess the potential of a new market and/or bar size, try building a population with zero generations. Because the initial population is randomly generated, the number of profitable strategies will give you an idea of how difficult it may be to find profitable strategies on the new data.
31. Always enter a value for trading costs or a reasonable bid/ask value. Otherwise, the resulting strategies may have too many unrealistically short (or even zero-length) trades.
32. The position sizing method and/or parameter values can be changed for an existing strategy by making new selections on the Position Sizing menu and re-evaluating the strategy from the Evaluation menu. Likewise, you can change the code type for an existing strategy by making a new selection on the Evaluation menu and re-evaluating the strategy by clicking the Evaluate button.

33. To copy the strategy code to the clipboard, right-click in the code window and select Copy Strategy, which selects and copies in one step.
34. The build process is designed to use all available cores on a multi-core computer. If you need to perform other tasks while Builder is running, try reducing the number of concurrent processes (Program Options, View menu). This will free up processing power for other applications.
35. If many of the strategies in the final population are the same, it means the results have converged. Use a Build Termination rule (GP Settings) to stop the build process based on convergence.
36. If the results displayed by Builder don't match the results you get when you run the strategy code in your platform, review the possible reasons outlined in Common Questions, Usage Topics (help files) and/or on the web site in the FAQ list under Support. It's usually something simple.
37. Simple strategies are sometimes more robust. To find simpler strategies, set the tree depth to zero. This will effectively remove the entry conditions, so that the only entry logic will be in the entry orders themselves.
38. To build more robust strategies, consider building over multiple price series. The different series could represent different markets or different bar sizes for the same market.
39. If you have past trading results for a successful strategy but you don't have the strategy logic, try using the performance results as build conditions and "reverse engineer" the strategy in Builder.

Chapter 8

Ribbon Menu

File Menu

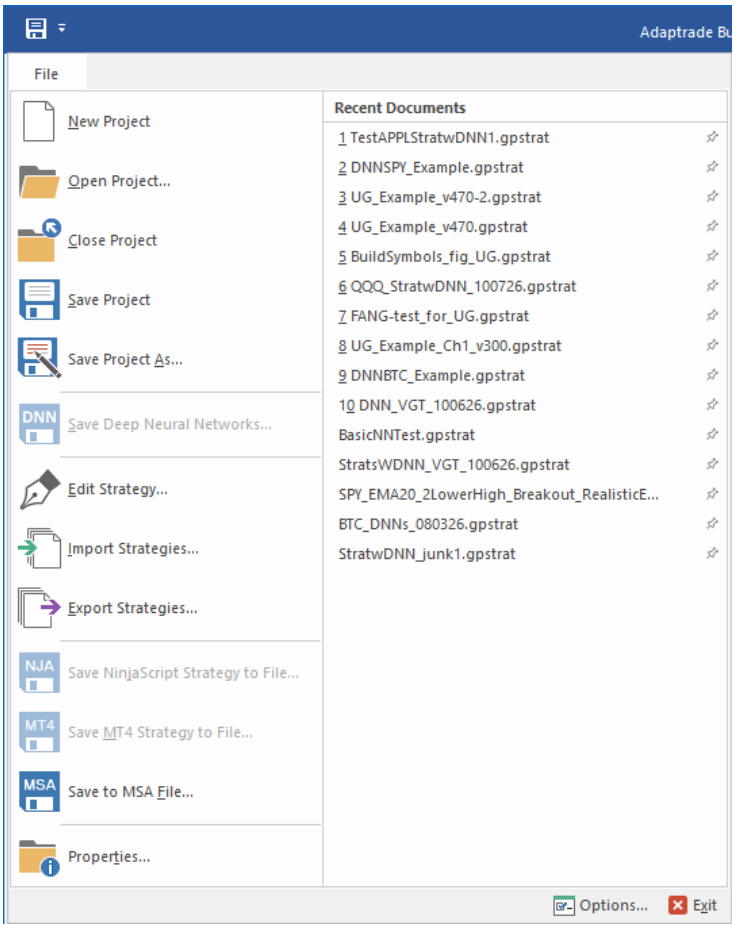


Figure 8.1. File menu commands.

The File menu is shown in Fig. 8.1. The file menu commands are listed below.

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.
Save Deep Neural Networks	Saves selected deep neural networks to DNN (.gpdnn) files.

Edit Strategy	Opens Adaptrade Editor with the currently selected strategy.
Import Strategies	Adds selected strategy files (.gpcode) to the current population.
Export Strategies	Saves selected strategies to strategy (.gpcode) files.
Save NinjaScript Strategy to File	Saves currently selected NinjaTrader strategy code to a file.
Save MT4 Strategy to File	Saves currently selected MetaTrader 4 strategy code to a file.
Save to MSA File	Saves trades and settings for currently selected strategy to a Market System Analyzer (MSA) file.
Properties	Display file properties.
Options	Use this command to customize the ribbon menu and the Quick Access Toolbar.
Exit	Exits Builder.

The Recent Documents list shows the 15 most recently opened project files. These project files can be opened by selecting them from the list.

If any command in the File menu is not applicable, the icon and associated text will be dimmed. For example, if no project file is open, the Close command will be dimmed. In Fig. 8.1, the commands for saving NinjaScript and MT4 strategies are dimmed because the selected strategy in the open project file is in EasyLanguage code. Most of the File menu commands will be dimmed 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 settings 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, it's necessary to open an existing project or create a new project before any further project work can be done in the program.

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**.

Save Deep Neural Networks Command

Use this command to save the deep neural networks (DNNs) currently selected in the Build Results table to files with the extension .gpdnn. Please see the description in the section Build Results Tables of the Build Results chapter.

Edit Strategy Command

Use this command to open Adaptrade Editor, the built-in strategy editor, with the currently selected strategy. The editor can edit any strategy in the Build Results tables and/or create new strategies, which can then be added to either population (main or Top Strategies). When saved, strategies created by Editor are written with the extension .gpcode. Please see the description of the **Edit Strategy** command in the section Build Results Tables of the Build Results chapter.

Import Strategies Command

Use this command to read in strategies from files with the extension .gpcode and add them to the end of the currently selected population of the Build Results table. Please see the description in the section Build Results Tables of the Build Results chapter.

Export Strategies Command

Use this command to save the strategies currently selected in the Build Results table to files with the extension .gpcode. Please see the description in the section Build Results Tables of the Build Results chapter.

Save NinjaScript Strategy to File Command

Use this command to save the currently selected NinjaTrader strategy code to a .cs file. The selected strategy must be in NinjaScript code format.

Save MT4 Strategy to File Command

Use this command to save the currently selected MetaTrader 4 strategy code to a .mq4 file. The selected strategy must be in MetaTrader 4 (MQL4) code format.

Save to MSA File Command

Use this command to save the trades and settings for the currently selected strategy code to a Market System Analyzer (MSA, extension .msa) file. Market System Analyzer (MSA) is position sizing and money management software for trading. Builder will prompt you to enter a name for the MSA file as well as a name for the text file to which the trades will be saved. The resulting .msa file can be opened directly by MSA version 3.3 or higher and will include all current Builder settings that are applicable to MSA. If more than one market was used in the build process, the point value and trading costs stored in the MSA file will be the ones for the currently selected market.

Properties Command

Use this command to display information about the currently open project file, including the file size and path, created and modified dates, date of last build, and the list of price files associated with the project, as in Fig. 8.2. The Builder File Version is the version of Builder that was last used to save the project file, which can be opened with that version or later of Adaptrade Builder. The Project Notes box allows you to enter notes describing the project.

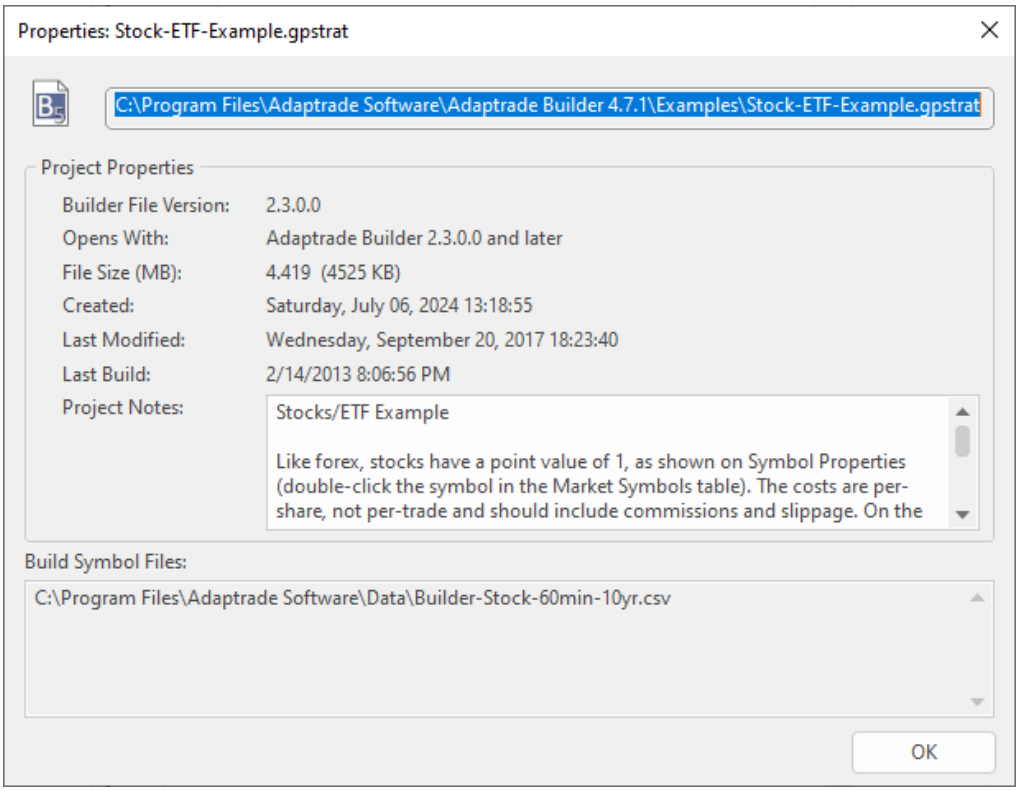


Figure 8.2. File Properties window.

Options Command

Use this command to add, remove, or rearrange items on the ribbon menu and to add or remove item from the Quick Access Toolbar (QAT). The QAT is the row of small icons in the upper left-hand corner of the title bar of the main frame window, above the ribbon. It is typically used to display icons for frequently used commands.

Exit Command

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

Build Menu

The Build menu is shown in Fig. 2.9 in Quick Start Steps in the Getting Started chapter. The settings and options of the Build menu were discussed throughout the Build Settings chapter.

Evaluation Menu

The Evaluation menu is shown in Fig. 2.10 in Quick Start Steps in the Getting Started chapter. The settings and options of the Evaluation menu were discussed in the Evaluation Options section of the Build Settings chapter.

Position Sizing Menu

The Position Sizing menu is shown in Fig. 2.11 in Quick Start Steps in the Getting Started chapter. The settings and options of the Position Sizing menu were discussed in the Position Sizing Settings section of the Build Settings chapter.

View Menu

The View menu, shown in Fig. 8.3, consists of three panels.

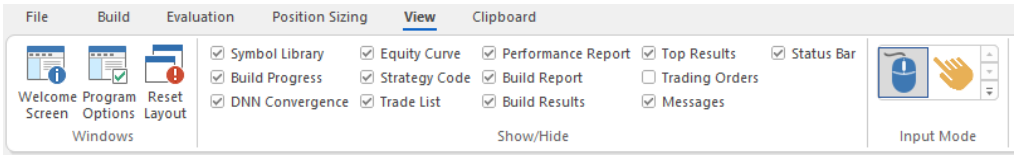


Figure 8.3. View menu of the ribbon.

Windows Panel

The **Welcome Screen** button displays a window that contains information to help you get started using the program. The Welcome screen is normally displayed when the program starts up. Options on the Welcome screen include buttons to open example files, a button to open the Quick Start Steps help topic, and tips for using the program. To prevent the screen from being displayed when Builder starts up, click the box "Don't display this screen at startup".

Clicking any of the three buttons for Example Projects will open the corresponding project (.gpstrat) file and display accompanying text in the text box. If Builder was installed in a folder other than the default presented during installation, it will be necessary to select the location of the example projects. By default, the example projects are stored in the Examples folder in the installation folder for Builder. The default path is C:\Program Files\Adaptrade Software\Adaptrade Builder x.x.x\Examples, where x.x.x is the version number (e.g., 4.0.0).

Clicking the View Next Tip button will randomly display one of approximately 35 tips for using Builder.

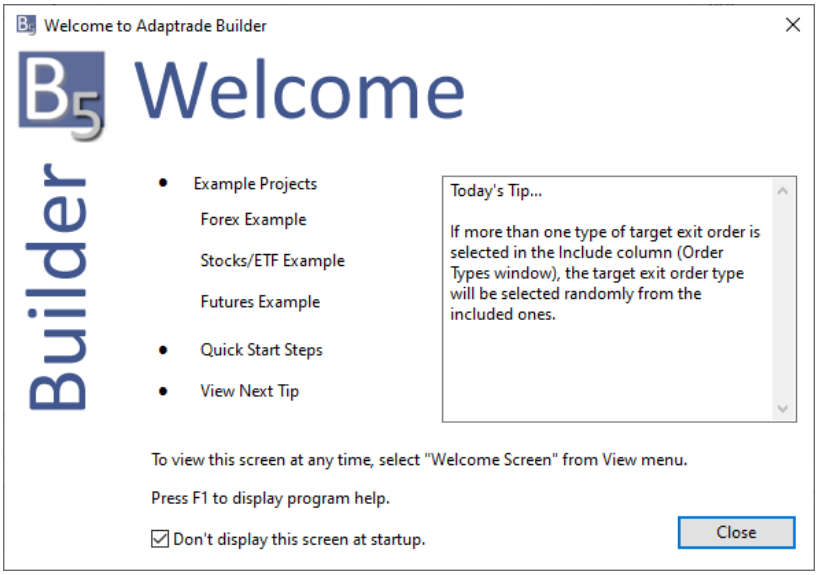


Figure 8.4. The Welcome screen displays instructional information.

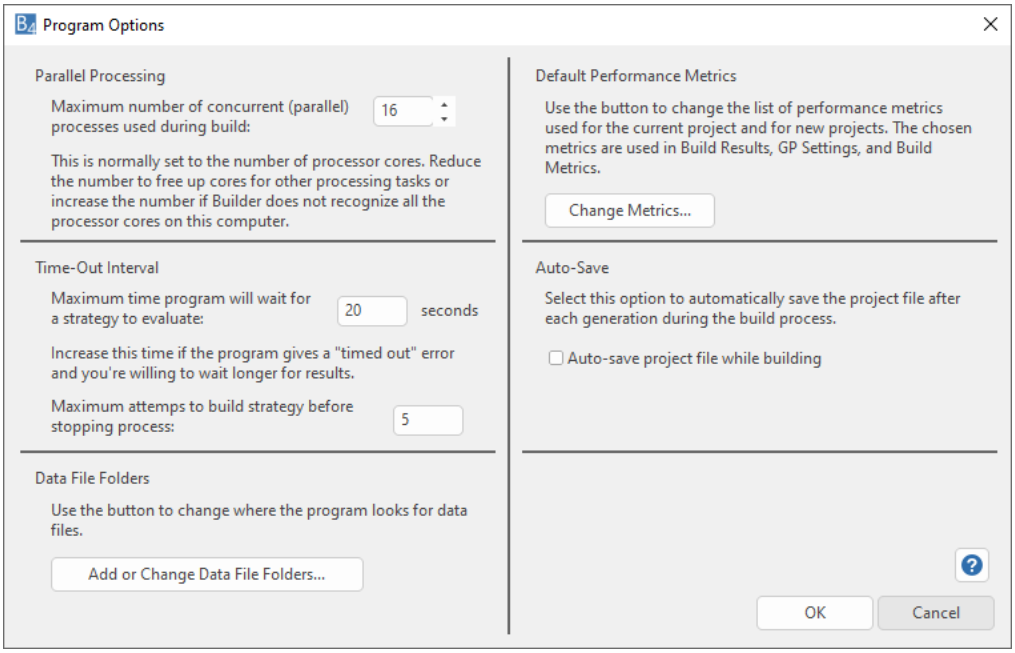


Figure 8.5. Program options are selected on the Program Options window.

The **Program Options** button opens the Program Options window, shown in Fig. 8.5, which allows the user to select program options. Options selected on this window are stored in the Windows registry and affect all project files.

There are currently five available program options. The first available option is the number of concurrent parallel processes used while building strategies. The default value is the number of available processor cores recognized by Builder. Set this to a lower number if you want to free up processing power for other programs running concurrently with Builder. Set this to a higher number if Builder has not recognized all the available cores on your computer, such as sometimes occurs with hyperthreaded processors.

The second option allows you to choose the set of performance metrics used with new projects. As explained in Build Metrics in the Build Settings chapter, the set of build metrics can be changed for any open project by right-clicking on the results table. The default set of build metrics, which is used for new projects, is selected here. Clicking the Change Metrics button opens the Change Performance Metrics window, as shown in Fig. 5.4. After modifying the list of selected metrics, click OK on the Program Options window to save the list in the Windows registry. The updated list will be used the next time a new project is created from the File menu.

The third option controls the length of the so-called time-out interval. This is the maximum length of time the program will spend to evaluate a single strategy during building and during the statistical significance test before terminating the process and displaying an error message. The default value is 20 seconds. You can also specify the number of retry attempts before the program will terminate the process. After an evaluation takes longer than the specified time-out interval, the strategy will be abandoned, and a new one will be created. This will be repeated for the maximum number of times specified here. If the program can't find a strategy that evaluates in less than the time-out interval in no more attempts than specified, the process (build process or significance test) will be stopped.

If it takes longer than 10 or 20 seconds to evaluate a strategy, there may be a problem with the build settings. For example, there may be too many trades in the strategy. If a strategy has hundreds of thousands of trades, it can sometimes take as long as a few minutes to evaluate. This sometimes happens if the trading costs have been set to zero. In this case, a strategy will not be penalized for numerous trades with zero profit/loss. Other reasons for exceeding the time-out interval include too much data or overly complex strategies. The time-out interval can be used to screen out strategies that have too many trades or are too complex. For example, if you set the time-out interval to five seconds, any strategy that takes longer than this to evaluate will time-out and be replaced. See Usage Topics, Build Time, for more on this topic.

The Auto-Save option, if selected, automatically saves the project (.gpstrat) file at each step during the build process. The project file is saved following population initialization and after each generation. The project file name must exist (i.e., the file must have been saved previously). A message is displayed in the Messages window during building to indicate that this option has been selected.

The Data File Folders option allows you to change where the program looks for price data files. Whenever a price data file is added to the Markets Symbols table on either the Build Symbols or Evaluation Symbols window, the path to the selected file is stored in the list shown here. When Builder reads the price data file, it looks for the price data file in one of the folders in the list. To view or modify this list, click the button "Add or Change Data File Folders". This will open the window shown below in Fig. 8.6.

If Builder cannot find a price data file shown in the Markets Symbols table, either re-select the file using the Symbol Properties window or select the file here using the Add Folder button. To remove a folder from the list, select the folder and click the Delete Folder button. When opening a price data file, Builder looks in the folders in the order they're listed in the Data File Folders window. Use the Move Up and Move Dn buttons to change the order of the folders if desired to put folders where your price data files are more commonly stored near the top of the list.

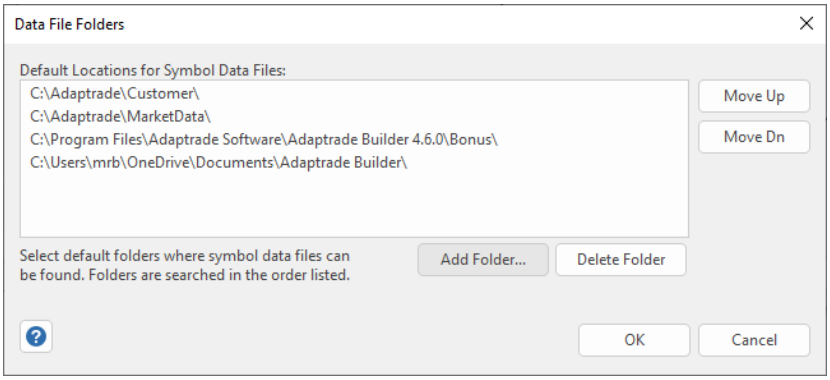


Figure 8.6. The Data File Folders window lists the folders where Builder will look for price data files.

The **Reset Layout** button returns the layout of the user interface, including the size and position of the various docking windows, to the default layout. Use this feature if the layout becomes altered and you're unable to restore it using the approach discussed in Program Layout and Configuration in the chapter Getting Started.

Show/Hide Panel
The Show/Hide panel displays panes and windows in the program that can be optionally opened or closed. Aside from the **Status Bar**, discussed below, each of these windows has been described elsewhere in the documentation. Click the check box to show or hide the window. If the box is checked, the window will be shown. Unchecking the box hides the window.

The **status bar** is displayed at the bottom of the main window. The left area of the status bar describes actions of right-click (context or popup) menu items as you move the mouse over them. On the right side is a Memory Usage bar, which shows the percentage of the computer's total available memory that is currently in use. Move the mouse over the indicator bar to see the numeric value.

Input Mode Panel
The Input Mode panel allows you to switch between mouse input and touch input. Click the mouse icon for normal input via a computer mouse. To use Builder with a touch-screen display, click the hand icon, which will expand the layout format of the ribbon to make it easier to select elements on a touch-screen display.

Clipboard Menu

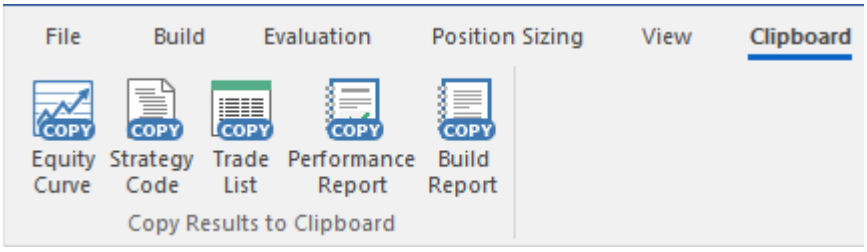


Figure 8.7. Clipboard menu of the ribbon.

The Clipboard menu, shown in Fig. 8.7, contains the following buttons to copy results for the currently selected strategy to the clipboard:

Equity Curve	Copy the equity curve as an image (bitmap) file to the clipboard.
Strategy Code	Copy the entire strategy code to the clipboard.
Trade List	Copy the list of trades from the Trade List window to the clipboard in tab-delimited format.
Performance Report	Copy the performance report to the clipboard in plain-text, tab-delimited format.
Build Report	Copy the build report to the clipboard in plain-text, tab-delimited format.

Ribbon Tab Buttons

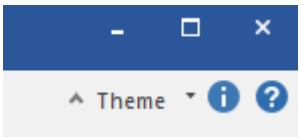


Figure 8.8. Ribbon tab buttons.

The ribbon tab buttons, shown in Fig. 8.8, are small buttons located in the upper right-hand corner of the ribbon. There are four ribbon tab buttons in Builder.

^ button	Minimizes or maximizes the ribbon menu to hide or show the contents of each menu.
Theme	Allows you to choose from among different user interface themes.
About Builder	Displays information about this application.
Help Topics	Offers you an index to topics on which you can get help.

Theme
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 theme from the popup menu. Once the theme has been changed, it will be stored locally, and the program will continue to use that theme until a new theme is selected.

About Builder
Use this command to display information about Adaptrade Builder, including the copyright notice, version number, recognized number of computer cores, and recognized amount of installed memory.

Help Topics
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.

Appendix: Performance Metrics

The different performance metrics calculated by Builder are described below. The first metrics listed below are used to assess strategy performance, while the last ones are used to assess the deep neural networks. All the metrics are reported in the Build Results tables and in the detailed Performance Report. The strategy performance metrics can be selected on the Build Metrics window, and the deep neural network metrics can be selected on the DNN Settings window. The first name shown below for each metric is the label displayed in the detailed Performance Report, followed by the abbreviation in parentheses, which is used on the Build Metrics window (for strategy metrics), or on the DNN Settings window (for DNN metrics), and in the results table. Unless otherwise noted, strategy metrics are calculated from the closed trades only. The metrics are listed below in the order in which they appear in the Performance Report.

Note: Any metric that depends on account equity values accounts for the external equity curves included in the evaluation, if any. This includes average profit/loss and return metrics and drawdown calculations.

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

Gross Profit. The sum of the winning trades.

Gross Loss. The sum of the losing trades.

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

Ratio L/S Net Profit (Ratio L/S Profit). Ratio of net profit for long trades to net profit for short trades.

Open Trade Profit/Loss (Open Profit). Net profit or loss for all trades that are open at the end of the trading period. There can be at most one open trade per market-system.

External Equity Curve Net Profit. Total net profit of all external equity curves over the trading period. This item is not displayed if there are no external equity curves. Because the external equity curve values are not dependent on the strategy results, there is no metric associated with this item.

Total Net Profit (Total Net Profit). The sum of the closed trade net profit, open trade profit/loss, and external equity curve net profit.

Trading Period. Time span of trading, as selected on the Build Symbols window.

Starting Account Equity. Value of account equity at the start of trading.

Highest Equity (Equity High). The highest value of account equity on an intra-bar basis over the trading period, including any external equity curves.

Lowest Equity (Equity Low). The lowest value of account equity on an intra-bar basis over the trading period, including any external equity curves.

Final Account Equity (Final Equity). Value of account equity at the end of the trading period, including any external equity curves.

Return on Starting Equity (Account Return). Percentage change in account equity from the starting account equity to the final account equity.

Number of Closed Trades (No. Trades). Total number of closed trades in the trading period.

Number of Winning Trades (No. Wins). Total number of trades with a profit/loss greater than zero.

Number of Losing Trades (No. Losses). Total number of trades with a profit/loss less than or equal to zero.

Trades Not Taken (No. Skipped). Total number of trades with a position size of zero.

Percent Profitable (Pct Wins). The number of winning trades expressed as a percentage of the number of trades with non-zero position size (i.e., excluding trades not taken).

Ratio L/S Number of Trades (Ratio L/S Trades). Ratio of the number of long trades to the number of short trades, excluding trades not taken.

Trades Per Year. Average number of trades per year, including open trades.

Trades Per Month. Average number of trades per month, including open trades.

Trades Per Week. Average number of trades per week, including open trades.

Trades Per Day. Average number of trades per day, including open trades.

Max Position Size (Max Shares). Largest number of shares or contracts for any one trade.

Minimum Position Size (Min Shares). Smallest number of shares or contracts of any trade, excluding trades with zero shares or contracts.

Average Position Size (Ave Shares). Average number of shares or contracts per trade, excluding trades with zero shares or contracts.

Largest Winning Trade (Max Win). Largest winning trade, where the winners are expressed in currency units (e.g., dollars).

Largest Winning Trade (%) (Max Win (%)). Largest winning trade, where each winner is expressed as a percentage of account equity at the time the trade was taken.

Average Winning Trade (Ave Win). Average of the winning trades, where the winners are expressed in currency units (e.g., dollars).

Average Winning Trade (%) (Ave Win (%)). Average of the winning trades, where each winner is expressed as a percentage of account equity at the time the trade was taken.

Average Bars in Wins (Ave Bars Wins). Average number of bars in winning trades.

Average Length of Wins (Ave Length, Wins). Average duration of winning trades in days, hours, minutes, seconds. In the Build Results tables and in the Build Metrics window, this is displayed as the number of days.

Max Number Consecutive Wins (Max Consec Wins). Largest number of winning trades in a row.

Average R-Multiple, Wins (Ave R-Mult, Wins). The R-multiple is the trade profit/loss divided by the risk of the trade, including trading costs. See Tharp, Van K. "Trade Your Way to Financial Freedom," 2nd ed., McGraw-Hill, New York, 2007. This metric is the average of the R-multiples for winning trades.

Largest Losing Trade (Max Loss). Largest losing trade, where the losses are expressed in currency units (e.g. dollars).

Largest Losing Trade (%) (Max Loss (%)). Largest losing trade, where each loss is expressed as a percentage of account equity at the time the trade was taken.

Average Losing Trade (Ave Loss). Average of the losing trades, where the losses are expressed in currency units (e.g. dollars).

Average Losing Trade (%) (Ave Loss (%)). Average of the losing trades, where each loss is expressed as a percentage of account equity at the time the trade was taken.

Average Bars in Losses (Ave Bars Loss). Average number of bars in losing trades.

Average Length of Losses (Ave Length, Losses). Average duration of losing trades in days, hours, minutes, seconds. In the Build Results tables and in the Build Metrics window, this is displayed as the number of days.

Max Number Consecutive Losses (Max Consec Losses). Largest number of losing trades in a row.

Average R-Multiple, Losses (Ave R-Mult, Losses). The R-multiple is the trade profit/loss divided by the risk of the trade, including trading costs. See Tharp, Van K. "Trade Your Way to Financial Freedom," 2nd ed., McGraw-Hill, New York, 2007. This metric is the average of the R-multiples for losing trades.

Average Trade (Expectation, Ave Trade). Sum of all trades, wins and losses, divided by the total number of trades, excluding trades with zero position size. 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.

Average Trade (%) (Ave Trade (%)). Average of all trades, where each trade is expressed as a percentage of account equity at the time the trade was taken.

Trade Standard Deviation (Trade Std Dev). Standard deviation of the trades, where the trades are expressed in currency units (e.g. dollars). The standard deviation is a measure of variability or dispersion. 68% of normally distributed values are within one standard deviation of the average. 99.7% are within three standard deviations of the average.

Trade Standard Deviation (%) (Trade Std Dev (%)). Standard deviation of the trades, where each trade is expressed as a percentage of account equity at the time the trade was taken.

Average Bars in Trades (Ave Bars). Average number of bars in all trades, both wins and losses. Consider setting a condition and/or objective for this metric if you want to limit the length of trades in the generated strategies.

Average Length of Trades (Ave Trade Length). Average duration of trades in days, hours, minutes, seconds for all closed trades, both wins and losses. In the Build Results tables and in the Build Metrics window, this is displayed as the number of days.

Min Bars in Trades (Min Bars). Minimum number of bars in all closed trades, both wins and losses. Consider setting a condition for this metric if you want to ensure the length of trades in the generated strategies is above a specified value.

Max Bars in Trades (Max Bars). Maximum number of bars in all trades, both wins and losses, including open trades. Consider setting a condition and/or objective for this metric if you want to limit the length of trades in the generated strategies.

Percent Time in Market (Time in Market (%)). Percentage of bars with a non-flat position, including open trades and excluding the MaxBarsBack number of bars.

Total Net Profit per Share (Total PL Per Share). Total net profit or loss per share, including open-trade profit/loss and deducting costs for closed trades.

Adjusted Net Profit per Share (Adjusted Total PL Per Share). Total Net Profit per Share divided by Percent Time in Market (expressed as a fraction). For example, if the total net profit per share is $100 and the time in market is 20%, the Adjusted Net Profit per Share would be $500.

Buy&Hold Profit per Share (Buy&Hold PL Per Share). Total net profit or loss per share assuming buying and holding throughout the period, excluding trading costs and excluding the MaxBarsBack number of bars.

Net Profit/Buy&Hold Ratio (PL/Buy&Hold Ratio). Ratio of the Total Net Profit per Share to the Buy&Hold Profit per Share.

Adjusted Net Profit/Buy&Hold Ratio (Adjusted PL/Buy&Hold Ratio). Ratio of the Adjusted Net Profit per Share to the Buy&Hold Profit per Share.

Average MAE (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.

Average MAE (%) (Ave MAE (%)). Average maximum adverse excursion (MAE), where the adverse excursion is expressed as a percentage of equity. The Ave MAE (%) is the average of the percentage 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.

Maximum MAE (Max MAE). Maximum value of the maximum adverse excursion (MAE). 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.

Maximum MAE (%) (Max MAE (%)). Maximum value of the maximum adverse excursion (MAE) in percent of equity. The Max MAE (%) is the largest of the percentage 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.

Average MFE (Ave MFE). Average maximum favorable excursion. The maximum favorable excursion (MFE) is the largest open-trade profit if one exists; i.e., the largest profit during the trade provided it has an open profit. The Ave MFE is the average of the MFE's over all trades, including open trades.

Average MFE (%) (Ave MFE (%)). Average maximum favorable excursion (MFE), where the favorable excursion is expressed as a percentage of equity. The Ave MFE (%) is the average of the percentage MFE's over all trades, including open trades.

Minimum MFE (Min MFE). Minimum value of the maximum favorable excursion (MFE). The Min MFE is the smallest of the MFE's over all closed trades.

Minimum MFE (%) (Min MFE (%)). Minimum value of the maximum favorable excursion (MFE) in percent of equity. The Min MFE (%) is the smallest of the percentage MFE's over all closed trades.

Maximum MFE (Max MFE). Maximum value of the maximum favorable excursion (MFE). The Max MFE is the largest of the MFE's over all trades, including open trades.

Maximum MFE (%) (Max MFE (%)). Maximum value of the maximum favorable excursion (MFE) in percent of equity. The Max MFE (%) is the largest of the percentage MFE's over all trades, including open trades.

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

Win/Loss Ratio (%). Ratio of average winning trade to average losing trade, where each trade is expressed as a percentage of account equity at the time the trade was taken.

Return/Drawdown Ratio (Ret/DD Ratio). Ratio of the percentage account return to the worst-case percentage drawdown.

Profit/Drawdown Ratio (PL/DD Ratio). Ratio of the Total Net Profit to the Worst-Case Drawdown. Both the net profit and drawdown are expressed in currency units (e.g., dollars).

Sharpe Ratio. Average monthly return minus the risk-free interest rate, all divided by the standard deviation of the monthly returns. The risk-free interest rate is assumed to be zero. This is a classic measure of risk-adjusted return. Higher values are better.

Sortino Ratio Average monthly return minus the target return, all divided by the so-called downside deviation. The downside deviation is similar to the standard deviation except that positive deviations are set to zero, whereas in the standard deviation, both positive and negative deviations are included. The Sortino ratio is an alternative to the Sharpe ratio and is sometimes preferred because it doesn't penalize positive returns that exceed the target value. In Builder, the target rate of return is set to zero. The Sortino ratio is generally higher than the Sharpe ratio.

MAR Ratio Average annual compounded rate of return divided by the maximum historical percentage drawdown. The MAR ratio is often used to evaluate and compare hedge funds and commodity trading advisors (CTAs). MAR ratios between 0.5 and 1.0 over long histories are generally considered excellent.

DoF Ratio. Degrees-of-freedom ratio. Number of closed trades divided by the product of 30 times the Complexity (number of strategy inputs). The DoF Ratio is based on a rule-of-thumb that says the number of trades should be at least 30 times the number of strategy inputs to avoid over-fitting; see, for example, "Be Your Own Hedge Fund", John Ehlers, Ric Way, Technical Analysis of Stocks & Commodities, Jan 2017, p. 16. Ratios over 1.0 are desirable.

Correlation Coefficient (Corr Coeff). Correlation coefficient of the equity curve. Also known as the Pearson's r formula. 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. If position sizing is applied such that larger positions are taken as the account equity grows (antimartingale position sizing), then the correlation coefficient is applied to the logarithm of the equity curve. Calculated from the detailed (bar-to-bar) equity curve, which includes external equity curves.

Trade Significance (Trade Sig). Student's t-test applied to the average trade. This can be used as a measure of strategy quality. It takes 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. Please refer to Evaluation Options in the Build Settings chapter for a discussion of the difference between this metric and the statistical significance test available on the Evaluation menu.

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 complexity is incremented by 2 to account for position sizing (1 for the method and 1 for the parameter value). 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 similar 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 similar 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.

Average Risk (Ave Risk). Average risk for all trades. The risk is the amount per contract or share that would be lost if the money management (protective) stop were hit, including trading costs.

Average Risk (%) (Ave Risk (%)). Average risk where the risk (see Ave Risk, above) is expressed as a percentage of account equity at the time the trade is taken.

Average R-Multiple (Ave R-Mult, Expectancy). The R-multiple is the trade profit/loss divided by the risk of the trade, including trading costs. See Tharp, Van K. “Trade Your Way

to Financial Freedom," 2nd ed., McGraw-Hill, New York, 2007. This metric is the average of the R-multiples for all trades, both wins and losses. Tharp refers to this as the expectancy.

R-Mulitple Standard Deviation (R-Mult Std Dev). Standard deviation of R-multiples over all trades, both wins and losses.

Total R (Total R-Mult). Total R-multiple. Sum of the R-multiples over all trades, both wins and losses. This is essentially the risk-normalized net profit.

Van Tharp SQN. SQN® is Van Tharp's System Quality Number; see for example, Tharp, Van K. "Van Tharp's Definitive Guide to Position Sizing," International Institute of Trading Mastery, Inc., North Carolina, 2008, p. 28. This metric is calculated as the Average R-Multiple divided by the R-Multiple Standard Deviation multiplied by the square root of the number of closed trades.

Note: For all R-Multiple calculations, for any trade for which the trade risk is undefined, the risk it taken as the average loss. The average loss is calculated over all segments, including both long and short trades.

Average Leverage (Ave Leverage). Average value of leverage over all positions. Leverage is defined as the position value divided by the account equity.

Maximum Leverage (Max Leverage). Largest value of leverage over all positions. Leverage is defined as the position value divided by the account equity.

Risk of Ruin. Probability of account depletion. See Vince, Ralph. "Portfolio Management Formulas," John Wiley & Sons, New York, 1990, p. 136, equation R3. The equation for risk of ruin assumes (1) unequal wins and losses, (2) a fixed number of contracts or shares per trade, and (3) ruin is defined as 100% account depletion.

Kelly f. The Kelly fraction (f) is typically used to calculate position size. However, here it's used as a measure of strategy quality. Expressed as a percentage, Kelly fractions greater than roughly 20% can be considered higher quality strategies. Please note: When setting a metric condition for Kelly f, it should be entered as a fraction even though it's displayed as a percentage.

Ideal Matches. The percentage correspondence between the set of ideal trades and the strategy trades. A metric value of 100 implies the same number of ideal trades as strategy trades and that each strategy trade matches an ideal trade. Note that for metric values other than 0 or 100, the value does not necessarily imply the percentage of trades (either strategy or ideal) that match one another. See Usage Topics, Ideal Trades for more information.

Average Entry Efficiency (Ave Entry Efficiency). Average efficiency of trade entries. For a long trade, the entry efficiency is calculated as: (maximum trade price - entry price) / (maximum trade price - minimum trade price), where the maximum and minimum trade prices are, respectively, the highest and lowest prices during the trade. For a short trade, the numerator is replaced by (entry price - minimum trade price). The Average Entry Efficiency is the average of the entry efficiencies over all closed trades. See Leo J. Zamansky, David C. Stendahl, "Evaluating System Efficiency", Technical Analysis of Stocks & Commodities, Oct 1997. Expressed as a percentage, the entry efficiency lies in the range of 0% to 100%.

Average Exit Efficiency (Ave Exit Efficiency). Average efficiency of trade exits. For a long trade, the exit efficiency is calculated as: (exit price - minimum trade price) / (maximum trade

price - minimum trade price). For a short trade, the numerator is replaced by (maximum trade price - exit price). The Average Exit Efficiency is the average of the exit efficiencies over all closed trades. Expressed as a percentage, the exit efficiency lies in the range of 0% to 100%.

Average Total Efficiency (Ave Total Efficiency). For a long trade, the total efficiency is calculated as: (exit price - entry price) / (maximum trade price - minimum trade price). For a short trade, the numerator is replaced by (entry price - exit price). The Average Total Efficiency is the average of the total efficiencies over all closed trades. Note: Average Total Efficiency = Average Entry Efficiency + Average Exit Efficiency - 1. This implies that if the sum of the entry and exit efficiencies is less than 1 (i.e., 100%), the total efficiency will be negative.

Average Annual Profit/loss (Ave Annual P/L). Average of all yearly profits (losses), including partial years.

Ave Annual Compounded Return (Ave Annual Return). Average annual rate of return assuming annual compounding.

Average Monthly Profit/loss (Ave Monthly P/L). Average of all monthly profits (losses), including partial months.

Ave Monthly Compounded Return (Ave Monthly Return). Average monthly rate of return assuming monthly compounding.

Average Weekly Profit/loss (Ave Weekly P/L). Average of all weekly profits (losses), including partial weeks.

Ave Weekly Compounded Return (Ave Weekly Return). Average weekly rate of return assuming weekly compounding.

Average Daily Profit/loss (Ave Daily P/L). Average of all daily profits (losses).

Ave Daily Compounded Return (Ave Daily Return). Average daily rate of return assuming daily compounding.

In Builder, **drawdown** is defined as a decline from the highest prior peak in intra-bar equity, including external equity curves, if any. The drawdown is measured from the highest prior peak to the lowest intra-bar equity value. Because Builder does not have access to intra-bar price points other than the open, high, low, and close, the order of the high and low prices is assumed based on the bar shape. This assumption can affect the intra-bar equity and drawdown calculations, which are therefore approximate on an intra-bar basis.

Number of Drawdowns (No. Drawdowns). This is the total number of drawdowns over the trading period.

Average Drawdown (Ave Drawdown). Average value of the drawdowns expressed in currency units (e.g., dollars).

Average Drawdown (%) (Ave Drawdown (%)). Average value of the drawdowns expressed as percentage declines from the prior highest peak in equity.

Average Length of Drawdowns (Ave Length, Drawdowns). Average duration of drawdowns in days, hours, minutes, and/or seconds. In the Build Results tables and in the Build Metrics window, this is displayed as the number of days.

Average Trades in Drawdowns (Ave Trades, Drawdowns). Average number of trades from the start of a drawdown until the first trade that makes a new equity high.

Worst-Case Drawdown (Drawdown). Largest drawdown, where the drawdowns are expressed in currency units (e.g., dollars).

Date at Trough (Date, Max Drawdown). The date at the lowest equity point during the worst-case drawdown.

Length of Drawdown (Length, Max Drawdown). Duration of worst-case drawdown, from the start of the drawdown to the trade that makes a new equity high. In the Build Results tables and in the Build Metrics window, this is displayed as the number of days.

Trades in Drawdown (Trades, Max Drawdown). The number of trades from the start of the worst-case drawdown until the trade the makes a new equity high.

Return on Maximum Drawdown (Return on DD). Total net profit divided by the worst-case drawdown, expressed as a percentage.

Worst-Case Drawdown (%) (Max Drawdown (%)). Largest value of the drawdowns expressed as percentage declines from the prior highest peak in equity.

Date at Trough (Date, Max % Drawdown). The date of the trade at the lowest equity point during the worst-case percentage drawdown.

Length of Drawdown (Length, Max % Drawdown). Duration of worst-case percentage drawdown, from the start of the drawdown to the trade that makes a new equity high. In the Build Results tables and in the Build Metrics window, this is displayed as the number of days.

Trades in Drawdown (Trades, Max % Drawdown). The number of trades from the start of the worst-case percentage drawdown until the trade that makes a new equity high.

Longest Drawdown. Duration of longest drawdown, from the first trade in the drawdown to the trade that makes a new equity high. In the Build Results tables and in the Build Metrics window, this is displayed as the number of days.

Start of Drawdown (Start, Longest Drawdown). Date at the start of the longest drawdown.

End of Drawdown (End, Longest Drawdown). Date at the end of the longest drawdown.

Percent of Equity (% of Equity, Longest Drawdown). Depth of longest drawdown, expressed as a percentage decline from the prior equity peak.

Longest Flat Period (Max Flat Period). Duration of longest period with no change in equity. In the Build Results tables and in the Build Metrics window, this is displayed as the number of days.

Deep Neural Network (DNN) Metrics

Mean Squared Error (Mean Sq Error). Sum of the squared differences between the actual price and the predicted price divided by the number of predictions, calculated using the

scaled prices. This is a common measure of training error when training DNNs that return a numerical value.

Net Points. Total number of price points that could be accumulated if you bought on the open of the next bar and sold on the close of the bar N bars in the future for an N-bar price prediction if the prediction is for a positive price change or sold short and bought back for a negative price change prediction. It's meant to give an idea of the profit potential of following the DNN's predictions.

Net Points per Bar (Net Point/Bar). Net Points divided by the number of predictions.

Percent Gain per Bar (Pct Gain/Bar). Net Points divided by the close at each accumulated "purchase", expressed as a percentage, and divided by the number of predictions. This can also be described as the average percentage gain and is effectively the percentage version of Net Points per Bar.

Gain Ratio. Percent Gain per Bar divided by the standard deviation of the percentage gains. This is similar in concept to the Sharpe ratio and can be effective as a build metric when a price series has a large historical price range, such as Bitcoin (BTC).

Direction Correct (%, Pct). Percentage of predictions that are directionally correct; e.g., if the price change is positive and the prediction is for a positive price change, the direction is correct.

Average Error (%, Pct). Average error in the price prediction, where each error is expressed as a percentage of price.

Average Error (Points, Pts). Average error in the price prediction in price points.

Maximum Error (%) (Max Error (Pct)). Largest error among all price predictions, where each error is expressed as a percentage of price.

Maximum Error (Points) (Max Error (Pts)). Largest error among all price predictions, where each error is in price points.

Minimum Error (%) (Min Error (Pct)). Smallest error among all price predictions, where each error is expressed as a percentage of price.

Minimum Error (Points) (Min Error (Pts)). Smallest error among all price predictions, where each error is in price points.

DNN Complexity. Total number of parameters among all DNN inputs, where each parameter corresponds to a strategy input, such as an indicator look-back length. Please refer to the description for the Complexity metric, above.

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 Parameter Ranges window.

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.

Weighted moving average

Weighted moving average of price. The weighted average assigns more weight to recent prices. The weight value decreases by 1 starting with the current bar. Because more recent prices are weighted more heavily, the weighted average is considered to be more responsive to price changes than the simple moving average.

Triangular moving average[1,2]

Triangular moving average of price. The triangular moving average is the simple moving average of the simple moving average of price, where the length of the simple moving averages is one-half the specified length of the triangular moving average. Because the triangular moving average is a double-smoothed simple moving average, it weights the middle portion of the look-back period most heavily.

Adaptive variable moving average (Adapt Variable MA)

This is an extension of Tushar Chande VIDYA indicator, which is essentially an exponential moving average in which the smoothing constant (alpha) adapts to market volatility or trend strength. In Chande's version, the effective period of the moving average decreases with the trend strength so that highly trending markets have a short moving average period and sideways or choppy markets have a long period moving average.

This version adds an extra input, TrendParam, which can be used to change the relationship between trend and moving average period. Positive values of TrendParam give the same relationship as in Chande's version, where a TrendParam of 1 is similar to VIDYA. Negative values of TrendParam reverse the relationship, so that the period increases with the trend strength and decreases with flat or trendless markets. A TrendParam of zero is a regular exponential moving average with no dependence on trend. Reasonable values of TrendParam are between roughly -5 and +5.

Zero-lag trend

Moving average of price based on John Ehlers' ITrend or InstTrend indicator in Cybernetic Analysis for Stocks and Futures, John Wiley & Sons, Inc, New Jersey, 2004, pp. 16, 24. According to Dr. Ehlers, this indicator has essentially zero lag for price frequencies within the range of interest to most traders.

Adaptive zero-lag trend (Adapt Zero-Lag Trend)
This is an extension of the zero-lag trend indicator, which is based on John Ehlers' ITrend or InstTrend indicator in Cybernetic Analysis for Stocks and Futures, John Wiley & Sons, Inc, New Jersey, 2004, pp. 16, 24.

In this version, the smoothing constant (alpha) adapts to market volatility or trend strength, depending on the input TrendParam, which defines the relationship between trend and moving average period. Positive values of TrendParam give a short moving average period during highly trending markets and a long period moving average during sideways or choppy markets. Negative values of TrendParam reverse the relationship, so that the period increases with the trend strength and decreases with flat or trendless markets. A TrendParam of zero gives a period equal to Length (i.e., independent of trend strength). The efficiency ratio is used as a measure of trend strength. Reasonable values of TrendParam are between roughly -5 and +5.

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

Triple exponential moving average (TRIX)[1]
The triple exponential moving average (TRIX) indicator starts by taking the natural logarithm of price, which is then smoothed by applying the exponential moving average three times. The TRIX is calculated as the difference between successive values of the triply smoothed result multiplied by a normalizing factor of 10000. The TRIX acts as an oscillator with values typically ranging from -100 to +100.

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)[1]
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[1,2]
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.

RocketRSI
John Ehlers' RocketRSI indicator calculates a smoothed version of an RSI scaled to [-1, +1] then applies the Fisher transform to convert the values to standard deviations. As with the standard RSI, 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. Reference: John Ehlers, "RocketRSI - A Solid Propellant For Your Rocket Science Trading", Technical Analysis of Stocks & Commodities, May 2018, V36:6 (8 - 12).

Inverse Fisher RSI
Inverse Fisher transform of the RSI indicator. The inverse Fisher transform provides sharper turning points for the RSI. The price values are smoothed with a 4-bar weighted moving average before applying the RSI. Indicator values lie between -1 and +1.

Inverse Fisher cycle
Inverse Fisher transform of John Ehlers' Stochastic Cyber Cycle indicator from Cybernetic Analysis for Stocks and Futures, John Wiley & Sons, Inc, New Jersey, 2004, p. 75. The Cyber Cycle is an oscillator derived from the cyclic component of price. After computing the stochastic cyber cycle and scaling it to -5 to +5, the inverse Fisher transform is taken to provide sharper turning points. Indicator values lie between -1 and +1.

Adaptive inverse Fisher RSI (Adapt Inv Fisher RSI)
This is an extension of the inverse Fisher RSI indicator, which applies the inverse Fisher transform to the RSI function to provide sharper turning points in the RSI. In this version, the smoothing constant (alpha) adapts to market volatility or trend strength, depending on the input TrendParam, which defines the relationship between trend and moving average period. Positive values of TrendParam give a short moving average period during highly trending markets and a long period moving average during sideways or choppy markets. Negative values of TrendParam reverse the relationship, so that the period increases with the trend strength and decreases with flat or trendless markets. A TrendParam of zero gives a period equal to Length (i.e., independent of trend strength). The efficiency ratio is used as a measure of trend strength. Reasonable values of TrendParam are between roughly -5 and +5. Indicator values lie between -1 and +1.

Adaptive inverse Fisher cycle (Adapt Inv Fisher Cycle)
This is an extension of the inverse Fisher cycle indicator, which is based on John Ehlers' Stochastic Cyber Cycle indicator from Cybernetic Analysis for Stocks and Futures, John Wiley & Sons, Inc, New Jersey, 2004, p. 75. The Cyber Cycle is an oscillator derived from the cyclic component of price. In this version, the smoothing constant (alpha) adapts to market volatility or trend strength, depending on the input TrendParam, which defines the relationship between trend and moving average period. Positive values of TrendParam give a short moving average period during highly trending markets and a long period moving average during sideways or choppy markets. Negative values of TrendParam reverse the relationship, so that the period increases with the trend strength and decreases with flat or trendless markets. A TrendParam of zero gives a period equal to Length (i.e., independent of trend strength). The efficiency ratio is used as a measure of trend strength. Reasonable values of TrendParam are between roughly -5 and +5. Indicator values lie between -1 and +1.

Commodity channel index (CCI)
The commodity channel index (CCI) is an oscillator calculated as the deviation of the average price, (H+L+C)/3, divided by the average deviation of the average price. The ratio is adjusted by a factor that helps keep most values within the range -200 to +200.

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)[1,2]
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.

Standard deviation
The standard deviation is the population (as opposed to sample) standard deviation of price over the past N bars, where N is an input. The standard deviation is a type of price difference and can be compared to other price differences within Builder.

Ulcer index
The Ulcer index calculates the average percentage drawdown in price, providing a general measure of downside risk in the market. See https://www.investopedia.com/terms/u/ulcerindex.asp.

Bollinger band
The Bollinger band is the average price plus a multiple of the standard deviation of price. If the number of standard deviations is positive, the result is an upper band. If the number of standard deviations is negative, the result is a lower band. The upper and lower bands are traditionally used to indicate a range of "normal" price movement.

Keltner channel[1]
The Keltner channel is the average price plus a multiple of the average true range. As with Bollinger bands, positive and negative multiples are used to produce upper and lower bands of the Keltner channel.

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. The use of this indicator in MetaTrader 4 strategies is not recommended because its value depends strongly on the starting bar, which cannot be controlled in MetaTrader. As a result, it's likely that different values for the indicator will be found in Builder than in MetaTrader 4.

Chaiken oscillator[1]
The Chaiken oscillator is the difference between two exponential moving averages of the accumulation distribution line.

Crosses Above/Below[1]
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)), or any other selected indicator that returns price, such as highest, lowest, and moving averages. 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.

Consecutive bars up (AS_ConsecBarsUp)
This indicator calculates the number of consecutive up prices, in which the "price" input can be any valid price, including any function that returns a price. If the price is lower, the count is reset. If the price is the same as the prior bar, the count is unchanged.

Consecutive bars down (AS_ConsecBarsDn)
This indicator calculates the number of consecutive down prices, in which the "price" input can be any valid price, including any function that returns a price. If the price is higher, the count is reset. If the price is the same as the prior bar, the count is unchanged.

Congestion count (AS_CongestCount)
The congestion count returns the number of consecutive bars where there is at least one price in common among all bars. High numbers indicate a "congested" or range-bound market.

Typical price
Average of high, low, and close; i.e., Typical price = (H + L + C)/3.

Floor trader pivots (AS_PivotS, AS_PivotR)
Floor trader pivots consist of three levels of support and three levels of resistance. With CP (central pivot) defined as the Typical Price of the prior bar, the floor trader pivots are defined as follows:

Support, level 1: 2 * CP - High[1]
Support, level 2: CP - (High[1] - Low[1])

Support, level 3: CP - 2 * (High[1] - Low[1])

Resistance, level 1: 2 * CP - Low[1]
Resistance, level 2: CP + High[1] - Low[1]
Resistance, level 3: CP + 2 * (High[1] - Low[1])

in which High[1] and Low[1] are the high and low price, respectively, on the prior bar.

Day of week

The day-of-week indicator returns an integer value from 0 to 6 to represent Sunday through Saturday, respectively.

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.

[1]Indicator not available for MetaTrader 4 code. [2]Indicator not available for AmiBroker code.

Appendix: Code Conventions

The strategy code generated by Builder uses various naming conventions for inputs, variables, and order type labels. Unless otherwise noted, the same naming conventions apply to all code types available in Builder.

The strategy code naming conventions for input variables are shown below.

N : integer or number, typically an indicator look-back length (e.g., N1, NATR).
X : floating point value (e.g., X1, X2, etc.).
En or Ent : entry (e.g., EntFr, ATRFrEn).
Ex : exit (e.g., TimeEx, NBarEx1).
Fr : fraction or multiple (e.g., EntFr).
Targ : target exit (e.g., TargFr).
MM : money management (protective) stop (e.g., MMFr).
StartEquity: starting equity value for position sizing calculations.
PSParam : position sizing parameter value, such a fixed fraction, delta (fixed ratio), percent of equity, etc.
RoundPS : true/false variable. True means round position size to nearest RoundTo increment.
RoundTo : value to round position size to if RoundPS is true; ignored if RoundPS is false.
SizeLimit : maximum allowable number of shares or contracts per trade.

As an example, consider the following code from the input section of a generated EasyLanguage strategy, with comments added to explain each input:

```
NBarEn1 (60),          // Number of bars, entry
NBarEn2 (14),          // Number of bars, entry
NBarEn3 (5),           // Number of bars, entry
EntFr (2.6909),        // Entry fraction of price difference or average true range (ATR)
NATR (30),             // Look-back length for ATR
TargFr (1.7681),       // Target fraction; fraction/multiple of ATR
MMFr  (1.51),          // Money management stop fraction; fraction/multiple of ATR
N1 (27),               // Indicator look-back length
N2 (59),               // Indicator look-back length
N3 (27),               // Indicator look-back length
N4 (59),               // Indicator look-back length
N5 (61),               // Indicator look-back length
X1 (10.0000),          // Indicator parameter (floating point) value
NATR (79),             // Look-back length of ATR
ATRFrEn (2.8858),      // Fraction/multiple of ATR for entry
TimeEx  (1015),        // Exit time
StartEquity (100000.00),   // Starting account equity; for position sizing calculations
PSParam (28.39),       // Position sizing parameter value; e.g., percent of equity
RoundPS (true),        // Round position size to nearest RoundTo
RoundTo (1),           // Round position sizing to nearest 1
SizeLimit (100);       // Limit position size to 100 shares or contracts
```

Strategy code generated by Builder also includes labels for the entry and exit orders. In TradeStation, these labels are listed in the Trade List and appear on the chart next to each entry and exit. The order labels added by Builder are shown below.

EnStop-L : Entry stop order
EnLimit-L : Limit entry order

EnMark-L : Market entry order
ExMark-L : Market exit order
ExTime-L : Exit-at-time order
ExNBars-L : Exit at N bars from entry order
ExStop-L : Money management (protective) stop exit order
ExTrail-L : Trailing stop exit order
ExTmRnge-L : Exit based on time range option (only for strategies built prior to version 1.3.0.0)
ExTarg-L : Target exit order

The "L" at the end of each label indicates that the order is for a long trade. Order labels for short trades end in "S".

Variables for Entry and Exit Conditions
The entry and exit conditions in Builder can be quite complex and may consist of multiple variables and multiple logical condition statements. Variables for numeric quantities, such as indicators, start with "Var" and end with "L" for long conditions or "S" for short-side conditions, followed by a number. Examples include VarL1, VarL2, VarS1, VarS2, etc.

Variables for logical conditions, which involve logical or inequality operators, follow the same convention as numeric variables except that they begin with "Cond". Examples include CondL1, CondL2, CondS1, CondS2, etc. An example showing how these variables are used is provided in Chapter 1, Entry and Exit Conditions.

The entry and exit conditions themselves are assigned to variables EntCondL, EntCondS, ExCondL, and ExCondS for, respectively, long and short entry conditions and long and short exit conditions.

Index

Adam....16, 83
AmiBroker ... 1, 2, 7, 13, 27, 38, 39, 101, 102, 119, 123, 151
artificial intelligence 3
Average Drawdown 143
Average Losing Trade....138
Average Number of Shares/Contracts....137
Average Number of Trades in Drawdowns....144
Average Trade....138
Average Winning Trade....137

back-test .. 39, 85, 86, 101, 102, 106, 107, 109, 119
build goals....5, 34
build set....5, 19, 75, 151
Build Symbols window....28, 32, 50, 53

charting 2, 27, 28
Clone Strategy command....98
complexity....77, 116, 141
crossover 3, 5, 7
Custom Indicators 47, 48

data mining....88, 114
data mining bias 114
data series.... 8, 32, 49, 50, 54, 55, 56
Data Window 30
deep neural network 13, 100, 103
deep neural networks ... 1, 6, 15, 30, 59, 65, 78, 81, 85, 92, 95, 99, 127, 129, 136
degrees-of-freedom 2, 141
delimiter 42, 45
DNNs 1, 2, 6, 15, 17, 30, 59, 65, 68, 74, 81, 82, 83, 84, 85, 92, 93, 95, 96, 99, 100, 129, 145
docking windows....92

EasyLanguage....iii, iv
editor....96, 129
Editor 80
equity curve symbol....44, 55, 102, 103
Equity Curve Symbol....42
error messages....94
ETFs....2
Evaluate command.... 58, 64, 73, 78, 85, 102
Evaluation Symbols window....28, 50, 53
Expert Advisor....39, 102
external equity curve.... 40, 42, 44, 50, 53, 57, 103, 136, 141

file properties....128, 130
Final Account Equity 137
fitness 3, 5, 34, 141
forex 2
futures....2

generations 2, 3, 5, 116, 141
genetic programming....2, 3, 4, 19, 34, 75
gradient descent....16, 83, 93
Gross Loss....136
Gross Profit 136

Highest Closed Trade Equity 136

ideal trades 60, 61, 62, 72, 110, 111, 112, 113, 114, 142
indicators....1, 2, 3, 5, 6, 75, 146, 150
Input Data....92
inputs....141
in-sample....2

Largest Losing Trade....138
Largest Winning Trade....137
Long/short symmetry 116
Lowest Closed Trade Equity 136
LSTM....15

main chart window....137, 143
Market Symbols table.....32, 42, 53, 55, 56, 57, 62, 77, 78, 85, 86, 99, 102
Market System Analyzer (MSA)....105
Max Number Consecutive Losses....138
Max Number Consecutive Wins 138
Max Number of Shares/Contracts 137
MaxBarsBack....38, 39, 101, 119, 120
maximum adverse excursion....139, 140
MetaTrader 4 1, 8, 26, 30, 38, 39, 42, 77, 101, 119, 121, 128, 129, 150
Minimum Number of Shares/Contracts....137
Monte Carlo 73, 98, 99, 100, 106, 108, 109
MultiCharts 1, 2, 6, 30, 38, 39, 42, 47, 49, 101, 119, 120
multi-core processors 117
mutation....3

neural network.....2, 5, 13, 14, 15, 18, 19, 26, 78
NinjaScript 101, 129

NinjaTrader 1, 2, 13, 26, 30, 42, 101
noise 2, 66, 114, 116
Number of Closed Trade Drawdowns 143
Number of Losing Trades 137
Number of Trades in Drawdown 144
Number of Winning Trades 137

optimal bar size 78
out-of-sample 2, 4, 106
over-fit 68, 93, 106, 107, 109, 114, 115, 116
over-fitting 2, 66, 106, 107, 109, 114, 115, 116, 141

panes 29, 92
performance metrics 5, 35, 136
population 2, 3, 5, 7, 94, 101, 102, 116, 117
position sizing 2, 37, 89, 90, 91, 105, 106, 130, 141, 152
price chart 28, 30, 38, 41, 47, 52, 57, 59, 60, 61, 62, 63
price data 3, 116
primary symbol 54, 55
processor cores 132

registry 29, 132, 133
results table 5
Return on Starting Equity 137
ribbon 28, 30, 63, 65, 128, 130, 131, 134, 135
robust strategy 106
robustness 106
rules 2, 3

sampling error 114
saved strategies 30, 117
secondary symbol 55
Semantic rules 8
SGD 16, 83, 85
Sharpe Ratio 140
signal 2, 4, 66, 114, 116
statistical inference 2
strategy code 30, 92, 101
strategy quality 88, 115, 141, 142
stress testing 73, 98, 99, 100, 106, 107, 109
symbol library 28, 40, 41, 49, 55, 56, 57
Syntactic rules 8

test segment 5, 33, 66, 67, 68, 70, 92, 93, 115
Trade Standard Deviation 138, 139
TradeStation ..... iii, iv, 1, 2, 6, 9, 26, 30, 38, 39, 42, 44, 47, 53, 77, 101, 118, 119, 120, 152
trading logic 3
Trading Orders 2, 62, 63, 64
trading platform 2, 39, 62, 76, 89, 101, 117, 119
training 2, 13, 68, 70, 92, 93, 99, 114, 115, 116
tree structure 6

under-fit 116

validation segment 2, 5, 33, 58
viable strategies 75

Win/Loss Ratio 140
Worst-case Drawdown 144