RunwiseFX EA

Rule Writing Guide

TABLE OF CONTENTS

1 INTRODUCTION

This document covers how write/amend trading rules for our EA that goes beyond what is supplied by default and in our library: https://www.runwisefx.com/library/
Our 'factory' supplied built-in rules will trade when configured indicators are in agreement and exit if go opposite. So, only if you need to go beyond this and cant' find what you need in our library then you will need this document.

Please note, the general user guide is available at:
http://www.runwisefx.com/runwiseFX_CSA_User_Guide.pdf
In this document we will assume you have read that.

1.1 Document Conventions

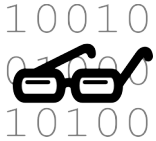
The following icons are used to throughout the document:

Take note – try to remember

Very import to take note of and remember - could cause undesirable results if ignored

Top tip – shortcut or other useful information that can make the system easier to use

Advanced feature/subject – can skip over when you are first learning how to use the system



Technical 'behind the scenes' detail that you don't strictly need to know but maybe interested in

2 CONFIGURATION FILE

The configuration file is an ordinary, human readable and editable text file (txt) and contains the indicators to capture, gui controls and associated trade rules for the EA to follow. When you use our CONFIG button pop-ups the changes made get save to this file.

★★★★★
top tip

We recommend using our CONFIG button pop-ups as much as possible to edit the configuration file, e.g. adding indicators via CONFIG | Indicator Values to Capture. However, when it comes to amending/adding trades rules or GUI panel controls then that will currently involve editing the configuration file. Note, we do have plans to expand our Windows application to allow the configuration file to edited via that.

The default name for the configuration the file is **runwiseFX_CSA_Supplied_Rules.txt** but different file names can be used by setting EA input Main_configFileName input. The file is located in the **MQL4\files** folder of your MetaTrader data folder (or MQL5\files on MetaTrader 5).

The file can be edited in a text editor, such as Notepad. The EA will auto re-read the file if it detects a change in the file date/time. Note, if any errors are detected by the EA then it will alerted with the line number in the file of where the problem is. If you wish to try the configuration in MetaTrader's Strategy Tester then recommend using the 'Copy Config to Strategy Tester' button in the CONFIG button pop-up.

2.1 Configuration File Format

The configuration file is divided into sections, where a section begins with […..] followed by lines that relate to section. For example, the section to configure the capture of a simple moving average would appear as follows:

```
[indi.sma]
indicatorName=F:MovingAverage
inputProperties=200,0,sma,close
colorIndex=
captureMode=0
shift=1
period=0
priority=2
nullValue=AUTO
minBars=
holdBars=
min=
max=
midPoint=
weighting=1
digits=AUTO
```

You will find the following sections in the configuration file:

[config]	Configuration items that broadly match the EAs inputs and are used to set thins like SL, TP, risk etc. Note, best to configure via CONFIG button on EA panel.
[indi.<refID>]	Indicator to be captured, where each indicator has a separate section. Can be configured via CONFIG | Indicator Values to Capture.
[gui.<type>.<refID>]	GUI control shown at the bottom of the EA panel. Again, each control has is a separate section.
[rule.<type>.<refID>]	Trading rule. Again, each rule has it's own section.

More details on these sections and how to configure them in is the rest of the document.

3 CAPTURING INDICATORS & CHART OBJECTS

The first aspect of automating a strategy is to capture the required indicator values. These can be from custom indicators installed on your MetaTrader, or built in MetaTrader functions for standard indicators, such as moving averages, stochastic, CCI, RSI, etc. It is also possible to capture values from objects that indicators place on the chart, i.e. as listed in **Charts | Objects** pop-up on MetaTrader.

Normally, the indicators don't need to be loaded on to the chart in order for the values to be captured. The exception is where capturing objects that indicators place on the chart. In this case, those indicators that create the objects will need to be present on the chart.

There are two ways of capturing indicators:

1) With Weighting box set which will cause the EA to derive the buy/sell indication from the indicator and keep a running total of whether buy or sell is being signalled for all the indicators configured with weighting set. Our trading rule supplied by default can then trade based on this total outcome.

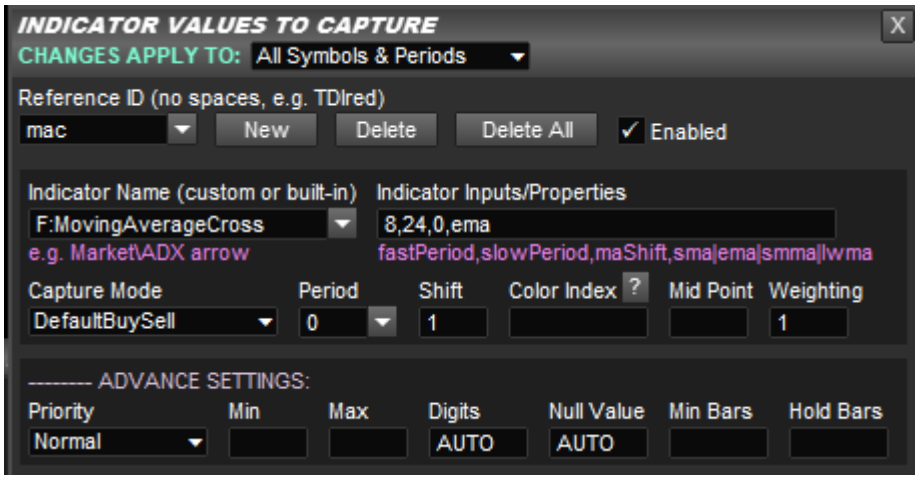
2) Where Weighting box is blank and the system just captures and records the raw indicator value which will need to referenced later in a trading rule. This gives you the maximum flexibility without the EA tyring to work out buy / sell itself. You can add a trading rule based on the indicators to do exactly what you want.

Lets look at these to ways in more detail an explain the differences.

3.1 Capturing Indicators with Weighting Box Set

This is the default way of capturing indicators. Here the Weighting box is set 1 and the system will capture the indicator value and keep a running total. This running total is then referenced in the trading rules supplied by default which will trade buy if all indicators are signalling buy and visa versa for sell.

For example here is how a moving average cross would be captured:

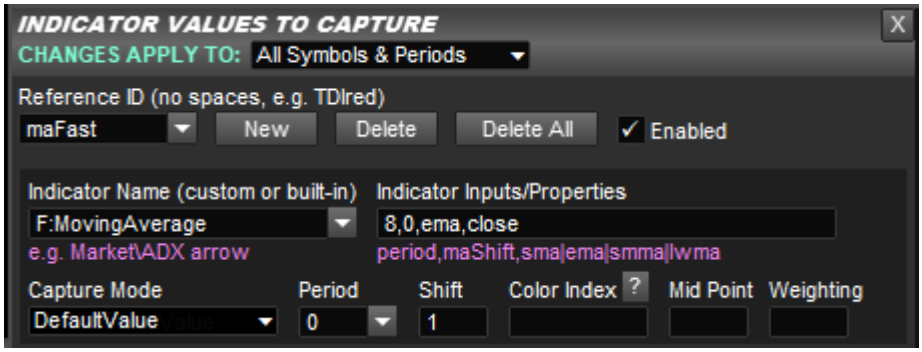


That's all you would need and our supplied entry rule would trade this based on the corss of the moving average, plus any other indicators configured.

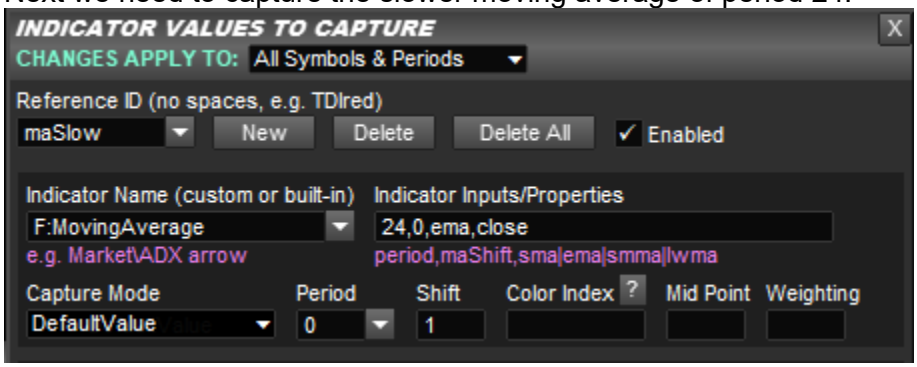
3.2 Capturing Indicators with Weighting Box Blank

Here is how a moving average cross would be configured if not relying on the system to derive the buy/sell indication. This approach gives maximum flexibility and we'll give an example of what we are taking about when we say that.

Firstly, would need to capture the fast moving average with a period of 8. Notice, the Capture Mode box is Default Value which just captures the raw value from the indicator, i.e. price from the moving average. Also, notice the Weighting box is blank.



Next we need to capture the slower moving average of period 24:



We would now need a trading rule to trade what has been captured. We explain this more later in the document, but would look like this in the configuration file:

```
[rule.entry.movingAverageEntry]
,i.maFast,/>,i.maSlow,TRADE_OPEN
```

The i.* allow the capture indicator values to be referenced. The > means greater than. Rules are evaluated for both buy and sell and the / before > means the > will be auto inverted to a < when evaluating for a sell. Finally, the TRADE_OPEN is the command which will signal trade open, subject to the Auto Mode configured in CONFIG | General Settings, which will either alert, set up pending order or open trade at market price, etc., on this trade signal.

Note, trade exit on opposite would need to configured separately. For example:

```
[rule.exit.movingAverageExit]
,i.maFast,/<,i.maSlow,TRADE_CLOSE
```

Of course, the burning question is why go to all of this trouble, as we showed something much simpler in 3.1. The answer is flexibility. Imagine you wanted not only the fast moving average to have crossed the slow one, but also by 5 pips. This could be achieved by adjusting the entry rule as follows:

```
[rule.entry.movingAverageEntry]
,f.pipDiffAuto({i.maFast};{i.maSlow}),>=,5,TRADE_OPEN
```

Here we are calling a function, which we explain later, that returns the number of pips between two supplied prices. Those prices are coming from the captured indicators. Note, we need { } around the indicator references in order to include them in function parameters. We call the { } an expression and will explain later. Finally, the 'auto' in the function name means it will auto invert when being evaluated for a sell and do slow – fast, so the >= 5 is still valid.

Finally, for completeness, there is a high chance you wouldn't want the 5 hard coded in the configuration file, but would want it on the panel, so could easily adjust. Here is how you would configure that. Again, this explained more in the later sections:

```
[gui.textBox.minPips]
caption=Min Pips:
defualtValue=5

[rule.entry.movingAverageEntry]
,f.pipDiffAuto({i.maFast};{i.maSlow}),>=,g.minPips,TRADE_OPEN
```

Here we have added a text box GUI control to the bottom of the EA's panel, with a default value of 5. We have then adjusted the entry rule to reference the value from this text box rather than hard coding the 5.

3.3 Direct Capturing of Chart Objects

Some indicators actually place objects on the chart rather than using color indexes. Note, only the `indicatorNameOnMetaTrader`, `inputProperties` and `digits` are used for the capture. These objects can captured as follows:

```
[indi.example]
indicatorNameOnMetaTrader= OBJ:<objectName>
colorIndex=-1
period=0
shift=0
inputProperties=<objectProperty>
interpretationMethod=A
interpretationParameters=
digits=AUTO
nullValue=AUTO
```

<objectName> is the object name as appears in **Charts | Objects** pop-up on MetaTrader.

<objectProperty> is property of the object to capture and can be one of the following:

a = arrow code

an = arrow code that is nearest to supplied shift value. Will search back through chart. Note, <objectName> only needs to contain part of the expected object name.

ad = arrow distance. Will search back through chart and return number of bars to supplied arrow object name. Can set shift to offset where to start search back from. Note, <objectName> only needs to contain part of the expected object name. Note, the distance is always measured from shift 0, i.e. 1 will be returned if arrow on closed candle even if supplied shift offset 1. So, shift 1 would just mean live candle is ignored when searching back, but doesn't offset returned value.

p = price 1

t = time 1 as number of seconds since 1st Jan 1970 (EPOC)

bgc = background color, in the format of R,G,B where each color is 0-255, e.g. 0,255,0

bgc2 = as background color but will attempt to convert to color constant, such clrOrange. For full list see:
https://www.mql5.com/en/docs/constants/objectconstants/webcolors

c = color, in the format of R,G,B where each color is 0-255, e.g. 0,255,0

c2 = as color but will attempt to convert to color constant

s = style

d = text/description

dn = as **d** but will only return number in object text/description, i.e. removes non-numeric characters

dt = as **d** but will apply a text trim which removes spaces before and after text

dtf = as **dt** but not necessary to supply exact <objectName> can be just part of it

t2 = time 2

t3 = time 3

p2 = price 2

p3 = price 3

Note, period is of no use as only objects that are shown on the current chart, i.e. current period, can be captured.

Sometimes object names that indicators place on charts can include the current chart symbol in that name. You can cater for this by using the expression {t.symbol} in the object name. The following example will successfully capture the text of an object that is named 'dfsrEURUSD57', where EURUSD is the current chart symbol:

```
[indi.dfsr]
indicatorNameOnMetaTrader=OBJ:dfsr{t.symbol}57
colorIndex=-1
period=0
shift=0
inputProperties=dt
interpretationMethod=A
interpretationParameters=
digits=0
nullValue=AUTO
```

interpretationParameters can be used to only return latter characters of text obtained from object. Use code ‘o’ to do offset, e.g. to ignore first character then set =o1
Note, the o is the letter o not number zero

4 GUI CONTROLS

The configuration can contain additional GUI items that appear in Rule Controls shown at the bottom of the EA's panel, i.e. labels, tick boxes, text boxes, list boxes and buttons. The rules can than refer to these GUI items and us the value they are set to, e.g. **g.TDIExit** would return 1 if TDIexit tick box is ticked, else 0 if not. The labels can be used by the rules to show information on the panel. Further, rules can triggered on the click of the GUI item, e.g. button press, these are called GUI rules and provide a script like behaviour, e.g. close all trades.

The different types of GUI items are configured as follows:

4.1.1 Label

[gui.label.<labelName>]
defaultValue=<value to appear on the label>
color=<color of label>

The **<labelName>** is name that can used in the rules to refer to the GUI item and should be unique and meaningful, e.g. totalProfit. The **defaultValue** is what the label text should be initially, but can be changed by the rules. **color** is the initial color of the label, bus also can be changed by the rules. If left blank then the default will be color used.

4.1.2 Tick Box

[gui.tickBox.<tickBoxName>]
caption=<text to appear next to the tick box>
defaultValue=<1 for the box to ticked by default, else 0 >

The **<tickBoxName>** is name that can used in the rules to refer to the GUI item and should be unique and meaningful, e.g. trendExitOn. Example:

```
[gui.tickBox.trendExitOn]
caption=Trend Exit
defaultValue=0
```

Creates a tick box with the caption 'Trend Exit', which is not ticked by default. The tick box can be referred to in the rules by **g.trendExitOn** and will have the value 1 if ticked else 0 if not ticked.

4.1.3 Text Box

[gui.textBox.<textBoxName>]
caption=<text to appear next to the text box>
defaultValue=<default text to place in the text box>

The **<textBoxName>** is name that can used in the rules to refer to the GUI item and should be unique and meaningful, e.g. pipTarget. Example:

```
[gui.textBox.pipTarget]
caption=Pips Target
defaultValue=40
```

Creates a text box with the caption ‘Pips Target’ with the default value of 50. The text box can be referred to in the rules by **g.pipTarget** and will have the value of the text contained in the box.

4.1.4 List Box

[gui.listBox.<listBoxName>]
caption=<text to appear next to the text box>
defaultValue=<default text to place in the text box>
selList=<comma separated list of items in list>

The **<listBoxName>** is name that can used in the rules to refer to the GUI item and should be unique and meaningful, e.g. pipTarget. Example:

```
[gui.listBox.timeframe]
caption=Timeframe:
defaultValue=M1
selList=M1,M5,M15
```

4.1.5 Button

[gui.button.<buttonName>]
defaultValue=<text to appear on the button>
direction=buy|sell

The **<buttonName>** is name that can used in the rules to refer to button and should be unique and meaningful, e.g. closeInProfitTrades. Example:

```
[gui.button.closeInProfitTrades]
defaultValue=Close In Profit Trades
```

Creates button with the text ‘Close In Profit Trades’.

Note, to add a rule that is executed when the button is pressed, the button name should be used in the rule header, for example:

```
[rule.gui.closeInProfitTrades]
,t.profit,>,0,FORCE_CLOSE
```

The direction is optional and if set will only execute the configure rule for that direction. This allows you to have a but for doing something related to a sell and another button for buy.

5 SPECIFYING RULES

Rule are specified by the [rule.*] section header followed by a series of lines (called logic lines). Each of the lines has several comma separated fields. The format is:

[rule.<ruleType>.ruleName]
<flags>,<operand1>,<operator>,<operand2>,<commands>,<failComment>
<flags>,<operand1>,<operator>,<operand2>,<commands>,<failComment>
<flags>,<operand1>,<operator>,<operand2>,<commands>,<failComment>
...

For example, an entry rule for the popular TDI indicator (Traders Dynamic Index) might be:

```
[rule.entry.TDIentry]
N,i.tdiGreen,/>,50,AND,"green not crossed 50"
N,i.tdiRed,/>,i.tdiYellow,TRADE_OPEN,"red not crossed yellow"
```

This will open a trade on the open of a new bar (signified by the flag N) when the TDI green line has crossed 50 and the TDI red line has crossed the TDI yellow line. Note, the '**/**' next to the greater than sign '**>**', is used to indicate that the '**>**' should be flipped to a '**<**' when evaluating the rule for a sell. The optional fail comment is used to report status back to the user (you) in the status text, i.e. whether it is the green or red line that is being waited for.

Full list of possible <ruleType> are as follows:

entry	Entry rules. Note, will be evaluated even when trade is open in case what to scale-in (add to position). You can use **t.tradeOpen** operand to know if trade is already open or not. The entry rules are evaluated on every tick from the market, i.e. will not be evaluated if market is closed.
exit	Exit rules. Only evaluated when trade is open in that direction and a tick has come in from the market.
gui	Evaluated when GUI item in Rule Controls is pressed/clicked, including list box selection. Or edit completed in a text box.
initGui	Evaluated when GUI is create or re-created, e.g. after configuration change, just prior to actually creating of GUI objects. Can be used to (say) create horizontal lines, or other objects, on the chart that other can rules use.
keypress	Evaluated on keyboard key press. Can be used to take action when a certain key is pressed. Use **t.keyPressCode** operand in order to know which key has been pressed. Note, you'll need the $ flag when evaluating lines containing text, see later.
postInitGui	Evaluated after GUI has been created/re-created. Can be used to initialise items after such an event.

timer	Evaluated on regular basis on timer ticks, even when the market is closed. The frequency in milliseconds will be determined by EA inputs Main_timerFrequencyMS*Main_timerRuleExecCount

Here is an example exit rule, again using the TDI indicator:

```
[rule.exit.TDIexit]
,i.tdiGreen,/<,50,TRADE_CLOSE,"green okay"
```

This will close the trade when the TDI green crosses back over 50. This the rule doesn't just operate on the open of new bar, but is evaluated on every tick. The optional fail comment will write on the status "green okay" if rule is active but green not yet crossed 50.

GUI rules and entry rules are evaluated in both directions (sell and buy). Hence, any TRADE_OPEN commands will open a trade in the direction that is being evaluated and for which the rules match.

The **<flags>** can be empty or one of **N,B,S,$,%**. N means evaluate line on a newly opened bar (candle). **B** means evaluate line when evaluating for buy, otherwise ignore the line. **S** means evaluate the line when evaluating for a sell, otherwise ignore the line. $ means evaluate as a string (text). **%** means evaluate as an integer (whole number), i.e. any floating point numbers will be rounded up.

The **<operator>** can be one of **=,<>,<,>,>=,<=,contains,notcontain**. The **contains** operator will look of operand 1 inside of another operand 2 and match if is there. The **notcontain** will match if the operand cannot be found inside operand 2. As mentioned in the examples, a '*!*' can precede the operator that will invert the operator when evaluating for a sell. This can avoid the need for separate lines with **B** and **S** flags.

A variety of operands can be used. Please contact us with your requirements.

A variety of commands are available, see 5.1. **<command>** can also be **AND** which means the following line(s) must match also, or **OR** which means either this line or the next one or one after if **OR** is still specified. Multiple commands can be specified separated by '|' (vertical bar). If a command is preceded by a '**!**' then the command is only executed if the expression does not match.

The **<failComment>** is optional and is displayed in the status lines on the control panel if rule fails to match on the line. It can be used to know why the rule is not performing the action associated with it. Note, if the comment contains a comma then don't forget to include double quotes around the text, or you may just wish to enclose double quotes around the fail comment for clarity.

Rules can have multiple lines that end in a command(s). However, the END_EVAL command can be used to stop the evaluation of the particular rule, regardless of what follows.

If any fields on rule logic line contain a **‘,’** (comma) then place double quotes around the field so that the parser doesn’t get confused, as commas are also used to delimitate fields.

5.1 Operands

Operands are things such as indicator values **(i.*)**, GUI control values **(g.*)** and items relating to the account/system **(a.*)** or a particular trade **(t.*)**. There is also a variety of use functions, e.g. add pips to price, maths - these are denoted by **(f.*)**. Finally and more advanced, there are variables local to rule **(l.*)** and **(v.*)** that are remembered.

When specifying operands in a logic line they can just be a value too.

The various types of operands are described in this section, including example usage.

5.1.1 Account/System (a.*)

These relate to the account or to the system/broker in general, e.g. current account equity, or current day of week, etc. Possible values are:

a.acctLev	Account leverage
a.acctNum	Account number
a.balance	Current account's balance
a.currency	Text label to describe the currency that the account is in. Can used visually or in alerts to display amounts nicely.
a.equity	Current account's equity. Example, close all trades on account if equity of account reaches 2500: `,a.equity,>=,2500,CLOSE_MULTI`
a.gmtDay	Day of month in GMT timezone, possible values are 1-31
a.gmtDOW	Day of week in GMT timezone, where 0 = Sunday, 1 = Monday etc. Example, only open trade if not a Monday: `,a.gmtDOW,<>,1,TRADE_OPEN`
a.gmtMin	Minutes of GMT time
a.gmtHr	Hours of GMT time
a.gmtOffset	Returns the current difference between GMT time and the local computer time in seconds, taking into account switch to winter or summer time. Depends on the time settings of your computer.
a.gmtSec	Seconds of GMT time
a.gmtYear	Year of GMT time
a.guiOn	Will return 1 if running in Strategy Test is visual mode
a.inHours	Will return 1 if current in configured trading hourse, else 0

a.isNotTesting	Will return 1 if not running in Strategy Tester, else 0
a.isTesting	Will return 1 if running in Strategy Tester, else 0
a.leverage	Account leverage. DEPRECIATED – please use a.acctLev instead
a.localDay	Day of month in local (computer) timezone, possible values are 1-31
a.localDOW	Day of week in local timezone, where 0 = Sunday, 1 = Monday etc.
a.localMin	Minutes of local time
a.localHr	Hours of local time
a.localSec	Seconds of local time
a.localYear	Year of local time
a.londonhr	House in London/UK timezone
a. mt4	Returns 1 if running a MetaTrader 4 system, or will return 0 if using MetaTrader 5.
a.numTickets	Current number of tickets opened on account
a.serverDay	Day of month in server (computer) timezone, possible values are 1-31
a.serverDOW	Day of week in server timezone, where 0 = Sunday, 1 = Monday etc.
a.serverMin	Minutes of server time
a.serverHr	Hours of server time
a.serverSec	Seconds of server time
a.serverYear	Year of server time
a.timestamp	Timestamp from server, as seconds from 1st Jan 1970 EPOC

5.1.2 Indicator Values (i.*)

This will access captured indicator values. Should be in format of i.<indicatorRef>

Example, open trade if TDI green value crosses 50, where TDI is green is being captured and stored in ref TDIgrn:

```
,i.TDIgrn,/>,50,TRADE_OPEN
```

Note, the / will invert if evaluating for a sell, i.e. > automatically becomes a <

Note, if the Weighting box is not blank then the system to derive the buy/sell indication from the indicator and return 100 for buy and -100 for sell. If you wish to access the raw indicator value with Weighting box set then use a ~ before the reference ID, e.g. i.~ma

5.1.3 Functions (f.*)

Functions are special in that they take parameters, which are specified in brackets after the function name and if more than one parameter then separated by a semi colon ;. Note, some parameters are optional and can be omitted to use default value, as described below.

Some functions contain the word 'auto' in their name. That means they will be auto inverted when being evaluated for a sell. A good example of this is f.pipsDiffAuto(<price1>;<price2>), which will return pips difference of price1-price2 when evaluating for a buy and give price2-price1 for sell. You'll find that this will invariably make writing rules easier without having a specify separate lines for buy and for sell.

Many functions have a 'period' parameter which refers to chart period. Can be 0 for current or something like M15. The 'shift' parameter refers to candle shift where 0 is live candle and 1 is candle just closed and 2 is candle before that and so on.

Provided functions are as follows:

f.anyGUIopen(name;value)	Used in multi-trade mode and return 1 if any trade index is open and has GUI item of specified *name* with a value equal to the specified *value*.
f.arPips(val)	Will normally just return supplied *val*. However, if % supplied with *val* then will return pips that represents that percentage of the currently daily range for the symbol the EA is trading on. For example, f.arPips(15) will just return 15. However f.arPips(15%) will return 15% of the daily range, which if say is a 120 pips will return 18.
f.char(ANSIcode)	Return character that represents supplied ANSI code. For example, f.char(65) will return A. For codes see: http://www.alanwood.net/demos/ansi.html
f.candleAvg(numCandles; period; shift)	Will examine chart and return the average number of pips between high and low of specified number candles, given by *numCandles*. Optionally, candle period and shift can be supplied for the operation. Note, if shift is not specified then will use 1.
f.iBodyAuto(period;shift)	Returns body high price if evaluating as buy and body lo price if evaluating as sell
f.iBodyHigh(period;shift)	Returns high price of candle body (not wick)
f.iBodyLow(period;shift)	Returns low price of candle body (not wick)

f.bodySize(period;shift)	Returns size of candle body (excluding wicks) in pips
f.candleAvg(period;shift; count)	Returns candle size/range, i.e. high – low, in pips. Can supply count which if greater than 1 will return average size for required number of bars/candles. If count is not supplied then will use 1, which will just be size of candle given by period and shift.
f.iClose(period;shift)	Returns candle close price
f.iHigh(period;shift)	Returns candle high price
f.iLow(period;shift)	Returns candle low price
f.iOpen(period;shift)	Returns candle open price
f.iTime(period;shift)	Returns candle time in seconds since EPOC
f.iVol(period;shift)	Returns candle volume
f.iLowset(period;shift; count)	Returns shift of highest candle high price. Will check back number of candles given by count. If count is omitted then will return highest price in the chart.
f.iHighest(period;shift; count)	Returns shift of lowest candle low price. Will check back number of candles given by count. If count is omitted then will return highest price in the chart.
f.iHighestClose(period;shift;count)	Returns highest candle close price. Will check back number of candles given by count. If count is omitted then will return highest price in the chart.
f.iLowestClose(period;shift; count)	Returns lowest candle close price. Will check back number of candles given by count. If count is omitted then will return highest price in the chart.
f.iHighestPrice(period;shift;count)	Returns highest candle high price. Will check back number of candles given by count. If count is omitted then will return highest price in the chart.
f.iLowestPrice(period;shift;count)	Returns lowest candle low price. Will check back number of candles given by count. If count is omitted then will return highest price in the chart.
f.pipDiffAuto(price1; price2)	Returns number of pips of price1-price2, if evaluating for buy and price2-price1 if evaluating for sell
f.mathAdd(val1;var2)	Returns val1 + val2
f.mathAddInt(val1;var2)	As f.mathAdd but assumes integers supplied
f.mathDiv(val1;var2)	Returns val1 / val2

f.mathMulti(val1;var2)	Returns val1 * val2
f.mathSub(val1;var2)	Returns val1 – val2
f.mathSubInt(val1;var2)	As f.mathSub but assumes integers supplied
f.priceAddPipsAuto(price;pips)	Returns supplied price adjust by supplied pips, which will be added if evaluating as a buy and subtracted if sell
f.priceAddPips (price;pips)	Returns supplied price with supplied pips added
f.priceSubPipsAuto(price;pips)	Returns supplied price adjust by supplied pips, which will be subtracted if evaluating as a buy and added if sell
f.priceSubPips (price;pips)	Returns supplied price with supplied pips subtracted
f.wickTop(period;shift)	Returns size of upper/top wick in pips
f.wickBottom(period; shift)	Returns size of lower/bottom wick in pips

5.1.4 Trade (t.*)

These relate to the trade, either open trade or trade potentially to be opened. Includes items related to market, e.g. ask and bid pricess Possible values are:

t.alertPrice	Price at which trade was signalled. Will be -1 if reset happened but not yet had signal. Will be 0 if reset not happened or trade voided.
t.ask	Current ask price
t.bid	Current bid price
t.buySell	Returns ‘buy’ of buy trade or being evaluated for buy and otherwise, ‘sell’
t.durationBars	How long trade has been opened for. If 0 the still in candle where trade was opened.
t.durationSecs	How long trade has been opened in seconds
t.peakPips	Peak pips profit trade has reached whilst been open
t.peakProfit	Peak profit amount trade has reached whilst been open
t.peakRR	Peak risk reward reached whilst trade has been open
t.profit	Trade profit amount, if trade open

t.profitPips	Trade profit in pips, if trade open
t.spread	Current spread in pips
t.tradeOpen	Returns 1 if trade open, else 0 if not open

5.2 Commands

The following commands can be used in rules and appear as the 4th field on the logic line of a rule. Remember multiple commands can be separated by '|' and if a command is preceded by '!' then it is only executed if the expression before it does not match, rather than match.

Parameters can use single quotes for strings. Parameters can also reference operands by enclosing them inside curly brackets {}. E.g.:
`ALERT('TDI yellow is now {i.tdiYellow}')`
This would raise an alert and text would include the value of tdiYellow indicator.

COMMAND & PARAMETERS	DESCRIPTION
ALERT(<msg>)	Will issue an alert with the text supplied in <msg>. Note, the alert will be push notified / emailed if the EA has been configured to do this via it's inputs. You must have logic in the rule to prevent repeated alerts that are not desired.
ALERT_LOCAL(<msg)	Similar to alert. However, we only be shown in the alert pop-up and not push notified or emailed, even if the EA has been configured to that.
AND	Not strictly a command but is used to join logic lines together. AND means the line and lines above must match the following lines (see OR as well).
CLEAR_BREAKEVEN	Will clear the breakeven tick box on the EA control panel for the direction being evaluated (see SET_BREAKEVEN).
END_EVAL(<failComment>)	Will stop the rule from being evaluated any further. An optional <failComment> will be shown on the status on the EA control panel, which is intended to indicate why the rule is not being evaluated any further.
FORCE_CLOSE(<prct>)	Will close any open trade with the specified percentage of the trade to close. If <prct> is not specified then the entire trade will closed, including any scale-in tickets. The command will execute as long as the EA is not set to manual mode for the direction (see TRADE_CLOSE that only has an effect if the EA is in alert/trade mode for the direction).
GUI_SET_LABEL(<name>;<text>; <color>;<size>)	Will set GUI control panel text label item with supplied text and optional color and font size.
GUI_SET_TEXTBOX(<name>;<value>)	Will set contents of GUI text box
MESSAGE_CONFIRM(<msg>)	Will pop-up a message dialog box showing the supplied <msg>. If the Cancel button is pressed then the rule will not be evaluated any further. Useful for confirming actions, e.g. button press, with the user.

	 Must only be used in GUI rules, as these are only executed on user click, i.e. when user is present. If the dialog box is not answered then the EA will be unable to process any further, including hidden stop loss, etc.
OR	Not strictly a command but is used to join logic lines together. OR means the current line or other lines above or below that also have OR, must match (see AND as well).
PLAY_SOUND(<wavFileName>)	Will play wav file located in Sounds folder of Data Folder. If no parameter is specified then any playing sound will be stopped.
PRINT(<msg>)	Will print, i.e. write, supplied message to log.
SCALE_IN(<prct>)	Will perform scale-in by either supplied <prct>, i.e. percentage of original trade – what has already been scaled in. Or, if <prct> is not specified then will scale-in based on corresponding setting on EA control panel.
SCREENSHOT	Will take screenshot of current chart that the EA is running on and place in the File folder of the Data Folder. Screen shot files are name rwfx_ss_* and can be controlled by **Gen_screenShot*** input properties.
SEND_MAIL(<subject>;<msg>)	Will email (if MetaTrader has been configured to do this) with the supplied subject and message.
SET_HIDDEN_SL(<price>; <mode>)	Will set hidden stop loss price and optionally stop loss mode. Note, hidden SL price only updated if mode is RULE or if a mode is specified, e.g. MAN
SET_HIDDEN_SL_MODE(<mode>)	Sets hidden stop loss mode
SET_HIDDEN_TP(<level>; <price>)	Will set hidden take profit (switch from auto to manual) of supplied level (number 1 upwards) and supplied price.
OBJ_CREATE(<name>;<type>)	Will create chart object of specified name and type. If <name> starts with '*' then the runwiseFX application ID will be inserted that allows the object to automatically deleted if the EA is removed from the chart. Note, initially objects are created with a location so they cannot be seen. TIP: Rule of type initGui can be used to create objects before the EA control panel is created, so they appear behind the control panel. <type> = object type **OBJ_*** with the OBJ_ omitted, e.g. HLINE will create a horizontal line object. A full list of object types can be found at: http://docs.mql4.com/constants/objectconstants/enum_object
OBJ_SET(<name>; <propertyIDValue>; <value>)	Will set object property to have supplied numerical value. The <propertyIDValue> is the second column of the table show here: http://docs.mql4.com/constants/objectconstants/enum_object_property. Note, other OBJ_SET_* commands are available that make it easier to

	set certain properties and are also required for text based properties.
OBJ_SET_COLOR(<name>;<color>; <style>)	Will set object color and optionally style. <color> can be RGB comma separated or one of web –color constants, e.g. clrMagenta, see: http://docs.mql4.com/constants/objectconstants/webcolors.  Don't forget if using commas in commands then will need to enclose the entire command field, of the logic line, in double quotes, as commas are also used to delimit each of the fields in the logic line. For <style> see http://book.mql4.com/appendix/styles for available styles, e.g. 0 = solid, 1 = dashed, etc.
OBJ_SET_FONT(<name>;<fontSize>; <fontName>)	Will set object font size and font name. If either parameter is blank (or in the case <fontName> not supplied) then no change will be made to that property.
OBJ_SET_POSITION(<name>;<x>;<y>; <corner>)	Will set position of the object on the chart with supplied X and Y co-ordinates and corner settings. Corner can be one of **LU** (left upper), **LL** (left lower), **RL** (right lower), **RU** (right upper). Note, if any parameter is blank then it will not be set, also <corner> may be omitted.
OBJ_ SET_PT(<name>; <price1>;<time1>; <price2>;<time2>;…)	Will set object price and time. This can be used for objects that are placed on the chart at a particular time and price. Note, <price2> and <time2> are optional. <time1> can also be omitted just to set price, e.g. for horizontal line. If <price1> is blank but <time1> is set then just TIME1 will be set, e.g. for vertical lines.
OBJ_SET_TEXT(<name>;<text>; <color>)	Will set the text property of the object with the supplied text and optional color (see OBJ_SET_COLOR for more details about setting object colors)
RESET_ALERT	Will reset alert price so that alert/trade is possible when indicators line-up. Otherwise, will need to get signal in opposite direction for reset.
PEND_CANCEL(["BOTH"])	Will clear pending order if set. If optional BOTH is supplied as a parameter then will clear both buy and sell pending orders in a single command.
SET_VAR(<name>; <value>;<symbol>; <temp>)	Will set a global variable of supplied name and value. Note, value has to be a number. Optionally <symbol> can also be specified to set of a specific symbol, which if '*' will cause the variable for all symbols to be set. If <symbol> = '..' then will replace with current symbol and period, concatenated together. If the optional <temp> is set as TEMP then the global variable will be marked as temporary, i.e. not preserved between closedown/restart of MetaTrader. Note, if name begins with # then variable stored within EA rather than using MetaTrader's global variables. **Note, currently if # used then rest of name needs to be a number 0-9.**
SEND_NOTIF(<msg>)	Will send push notification email (if MetaTrader has been configured to

	do this) with the supplied message.
SET_BREAKEVEN	Will set the breakeven tick box on the EA control panel for the direction being evaluated. This command should only be issued when a trade is open and in profit – if not in profit then would immediately exit the trade (see CLEAR_BREAKEVEN).
SET_PEND(<price>; <action>)	Set pending line for the direction being evaluated, including setting type and mode. <action> as shown in Pend: pop-up, e.g. TT. Note, the limit/stop type will be set depending on where the line is placed in relation to the price. However, if the trade is open then the type will be set to scale-in, provided the price is appropriate for a scale-ion, e.g. if buy then price needs to be above current price. The <action> can be omitted to leave as is.
SET_PEND_PRICE(<price>)	Set the pending line price for the direction being evaluated. However, the price will not be updated if it causes the pending line to jump over current price, e.g. from Limit to Stop type, i.e. is only intended to update the price not the type (Limit/Stop).
TRADE_CLOSE(<prct>)	Will close any open trade (or alert if alert mode is set) with the specified percentage of the trade to close, if the mode is set to trade. If <prct> is not specified then the entire trade will closed, including any scale-in tickets. See FORCE_CLOSE that will work regardless of mode setting unless the mode is manual.
TRADE_OPEN(<riskAdjustPrct>; <alertText>)	Will open a trade in the direction that is being evaluated using the settings on the EA control panel, provided mode is set to trade. Will alert if mode is set to alert. If a trade is already open then no action will be taken. Note, if you want to open additional tickets then use the SCALE_IN command. The optional <riskAdjust> can be used to adjust the risk that has been specified on the EA control panel, e.g. if risk is 3% of account but this parameter is 10, i.e. 10%, then the risk used will be 3.3%. Negative values can be used to reduce risk. This parameter allows (for example) rules to adjust risk based the probability of the trade being successful. The optional <alertText> will replace the standard text used when the EA is in alert mode rather than trade mode.
VOID_ALERT	Sets the alert price 0, which means won't open a trade even if TRADE_OPEN command is run (will just show as 'stale'). Will have to wait to next time alert is reset. Generally, this is when the market is in the opposite direction.

5.3 Expressions {….}

It's possible to use any operands or functions inside parameters or such things as alert text. You just need to enclose in curly brackets. For example:

`f.priceAddPips({t.bid};{g.pips})` - will return the current bid price with value from GUI control to supply pips to add

Be careful if using nested parameters, where need to use single quotes so that will be parsed correctly. For example,

`OBJ_SET_PT(*confirmPrice;'{f.priceAddPipsAuto({g.breakPrice};7)}')` – will set confirmPrice chart object with value from GUI control breakPrice offset by 7 pips

6 CONTACTS

Runwise Limited
The Tramshed
Walcot Street
Bath
BA1 5BD
United Kingdom

Email: support@runwisefx.com
Web: www.runwisefx.com

END OF DOCUMENT