Roolo dynamic prices formula guide

If setting a fixed price on your advertisement does not give you enough control, then choosing the Dynamic Price with Formula option will allow you to enter your own equation for calculating a dynamic price.

How it works

The Roolo systems collect pricing information from the top exchanges around the world and use this information to constantly update the price shown against your advertisements.

Each market value that we collect is represented by a placeholder name, such as **bitcoinaveragegbp_mid**. Every time that these values are updated, they are inserted in to the formula that you entered and a current live price is determined.

Your formulas may contain certain operators and functions, described below, which allow you to specify an advertisement price relative to, or as a function of, any combination of the values that we collect.

Operators and functions

Arithmetic Operators

+	Addition.
-	Subtraction The – operator may also be used as a negation operator : e.g. 6 + (-2) which would equal 4.
*	Multiplication
/	Division. Note that the result of a division operation results in a number that may have have a fractional part.
^	The ^ operator raises one number to the power of another. For example 2^3 evaluates to 8 (i.e. 2*2*2).

Logical operators

All of the logical operators compare two values and return 0 (Zero) if the comparison evaluates to false or 1 (One) if the comparison evaluates to true. These operators are intended for use with the 'If' function.

<	Less-Than. Is 'true' if the first operand evaluates to a smaller number than the second operand. E.g. 4 < 8 would evaluate to 1 (True).
<=	Less-Than or Equal to.
>	Greater-Than.
>=	Greater-Than or Equal to.
=, ==	Is-Equal-To. Both of these operators make a simple comparison, the == operator is included for the comfort of developers used to programming languages that use this convention.
!=, <>	Is-Not-Equal-To. E.g. 5 != 5 returns 0 (False) whilst 42 != 3 returns 1 (True).

Functions

Functions, expressed as a name followed by a pair of round brackets which may contain one or more comma separated values, are intended to simplify complex calculations and to enable you to construct more intelligent price formulae.
Each function parameter may be made up of other price values, operators or functions.
Please note that there must not be a space between the function name and the first opening bracket.

sin, cos, tan	Standard Trigonometric functions. Each of these functions takes a single parameter which should evaluate to a number expressed in radians. The constant 'pi' is defined to make trigonomic calculations easier.
asin, acos, atan	Trigonometric functions, taking a single parameter, returning a value in radians.
sqrt	Returns the square root of the single parameter given to it. E.g. sqrt(9) evaluates to 3. Sqrt(4*4*4*4) evaluates to 16.
abs	Absolute. Returns the absolute (i.e. positive) value of the single parameter given to it.
ln, log	Natural Logarithm
trunc	Returns the integer part of the number given to it. E.g. trunk(3.65) returns 3.
ceil	Returns the next highest integer value by rounding up the value passed to it if necessary. E.g. ceil(1.1) returns 2 whilst ceil(-5.5) returns -5.

floor	Returns the next lowest integer value by rounding down the value passed to it if necessary. E.g. floor(1.1) returns 1 whilst floor(-5.5) returns -6.
round	Rounds the given value to the nearest integer. .5 values round up.
sgn	For a given value, returns (-1) if negative, 1 if positive or 0 if the parameter value is zero.
max	This function takes an unlimited number of parameters and will return the largest of the given values. E.g. max(5,66) returns 66, max(1,33,77,3,56) returns 77.
min	This function takes an unlimited number of parameters and will return the smallest of the given values. E.g. max(5,66) returns 5, max(1,33,77,3,56) returns 1.
average	This function takes an unlimited number of parameters and will return the average of the given values. E.g. average(5,66) returns 35.5, average(1,33,77,3,56) returns 34.
if	This function takes three parameters : If(condition, trueValue, falseValue) If the condition value contains a non-zero (i.e. true) value, then the second 'trueValue' parameter is returned, otherwise the third 'falseValue' parameter is returned. This enables you to create conditional logic and the functions can, of course, be nested. E.g. if(5 < 6, 7,8) returns 7 because 5 is less than 6 so the second parameter is returned.
or	Returns 1 (true) if any of the given parameters is true (i.e. not zero), otherwise it return 0 (false). E.g. or(0, 5, 77, 0, 3) returns 1 (true).
and	Returns 1 (true) only if ALL of the given parameters are true (i.e. not zero), otherwise it return 0 (false). E.g. and(1,2,3) returns 1 (true) whilst and(1,2,3,0) returns 0 (false).
not	This function takes a single parameter and returns 1 (true) if the given value is 0 (false), or 0 (false) if the given value is not zero (true).
fx	Fx(FromPrice, FromCurrency, ToCurrency) This function is provided as a convenience for converting prices from one currency to another. The first parameter is the price to convert, the second parameter is the currency in which that price is denominated, the third parameter is the desired currency of

denomination. The Third parameter can be omitted, in which case a price to USD is calculated.
E.g. Fx(100, GBP, EUR) would return the value of 100GBP converted to Euros.
Fx(bitstampusd_bid, USD, GBP) would return the current BitStamp USD Bid price, converted from US Dollars to British Pounds.

Placeholder names (variables)

As well as fixed numbers and functions, your formulae can contain placeholders representing prices published by other markets. These placeholders can be used anywhere in your formula and a new price will be calculated each time any of them change.
The prices that we collect are all taken from the public APIs of each site.
The update frequency and availability of these prices is subject to the business practice and good will of the respective exchanges.

All of the placeholder prices are in the form :

<Exchange><Currency>_<price>

For example, Bitfinex USD Bid price has the placeholder : bitfinexusd_bid
The placeholders are not case sensitive.

Placeholders :

Exchange	Currency(s)	Price(s)	Example(s)
Bitfinex	USD	Bid, Ask, Last High, Low, Mid	bitfinexusd_bid bitfinexusd_mid
Cryptopay	EUR GBP USD	Bid, Ask, Last High, Low, Mid	cryptopayeur_bid cryptopayusd_high
BTC China	CNY	Bid, Ask, Last High, Low, Last Mid, VWAP	btcchinacny_last btcchinacny_mid
BTCe	EUR RUR USD	Bid, Ask, Last High, Low, Last Avg, Mid	btceeur_avg btceusd_last
Coinbase	USD	Bid, Ask	coinbaseusd_bid coinbaseusd_ask
Bitbargain	GBP	Avg_1h Avg_3h Avg_6h	bitbargaingbp_avg_1h bitbargaingbp_avg_12h

		Avg_12h Avg_24h	
Igot	AED AUD EUR GBP HKD INR INR_India KES NZD SGD USD	Open High Low Mid	igotusd_mid igothkd_open
BitStamp	USD	Bid, Ask, Last High, Low, Mid	bitstampusd_bid bitstampusd_ask
OKCoin	CNY	Bid, Ask, Last High, Low, Mid	okcoincny_ask okcoincny_low
Huobi	CNY	Bid, Ask, Last High, Low, Mid	huobicny_high huobicny_low
Bitcoin Average	AUD, BRL CAD, CHF CNY, EUR GBP, IDR ILS, MXN NOK, NZD PLN, RON RUB, SEK SGD, USD ZAR	Bid, Ask, Last Avg, Mid	bitcoinaverageaud_mid bitcoinaverageusd_last
Local Bitcoins (localbtc)	AED, ARS, AUD BRL, CAD, CHF CLP, CNY, COP CZK, DKK, EUR GBP, HKD, HRK HUF, INR, IRR JMD, KES, MXN MYR, NGN, NOK NZD, PEN, PHP PLN, RON, RUB SAR, SEK, SGD THB, TRY, UAH USD, VEF, XAF ZAR	LAST AVG1 AVG12 AVG24	localbtcusd_last localbtczar_avg1
ANX	CAD EUR GBP HKD JPY NZD SGD USD	Bid, Ask, Last High, Low, Mid Avg, Vwap	anxcad_vwap anxcad_low

Kraken	EUR GBP JPY USD	Bid, Ask, Last High, High24, Low, Low24, Mid Vwap, Vwap24	krakengbp_mid krakenjpy_last

More Placeholders :

In addition to the market price placeholders, there are a number of system names that are there to help you further customise your prices.

Pi, e	Standard mathematical constants
timestamp	Returns the current time measured as seconds since 1 Jan 1970. This is a Unix convention and equates to GMT.
Year	Four digit year of the timestamp date
Month	Month of the timestamp date, an integer between 1 and 12.
Day	Day of the timestamp date, an integer between 1 and 31.
Hour	Hour of the timestamp date, an integer between 0 and 24.
Minute	Minute of the timestamp date, an integer between 0 and 59.
Second	Second of the timestamp date, an integer between 0 and 59.

These values can be used with the logical operators and the IF function to, for example, adjust your prices when outside of normal hours.

Currency (FX Rate) placeholders.

There are placeholders for many international currencies using the standard ISO currency codes including, but not limited to :

AED, ARS, AUD, BRL, CAD, CHF, CLP, CNY, COP, CZK, DKK, EUR, GBP, HKD, HRK, HUF, IDR, INR, IRR, JMD, KES, MXN, MYR, NGN, NOK, NZD, PEN, PHP, PLN, RON, RUB, SAR, SEK, SGD, THB, TRY, UAH, USD, VEF, XAF, ZAR

These placeholders represent the amount of local currency that equals 1 USD, so to convert a price from Hong Kong Dollars to US Dollars, for example, the formula might be :

SomeHKDPrice / HKD

Which would result in the USD equivalent of the HK Dollar price. To convert the same price to GBP, you would need to convert to USD as above and then into Pounds Sterling:

(SomeHKDPrice / HKD) * GBP
or the equivalent : SomeHKDPrice * (GBP/HKD)
or using the FX function : fx(SomeHKDPrice, HKD, GBP)

More Examples

To calculate the Average mid price of Bitstamp, BitFinex and Kraken USD prices :

```
Average(bitstampusd_mid, bitfinexusd_mid, krakenusd_mid)
```

To return that as a GBP Price :

```
Average(bitstampusd_mid, bitfinexusd_mid, krakenusd_mid) / GBP
```

Or

```
Fx(Average(bitstampusd_mid, bitfinexusd_mid, krakenusd_mid), USD, GBP)
```

To price off the BitcoinAverage GBP Mid price and adjust your price by +5 if the time (GMT) is before 9AM or After 6PM :

```
bitcoinaveragegbp_mid + if(or(hour < 9, hour >= 18), 5, 0)
```