Chartboard

Expert Advisor Quick Reference

Ivyware

Melbourne, Victoria, Australia, 3130
Phone (+61) (0) 417-337-313
www.ivyware.com.au

9th February, 2018

Python extension functions & classes

Chartboard supports the development of custom Expert Advisor scripts via Python 3.6 extension functions and classes. These scripting facilities have been integrated into the Workspace environment providing the following functionality.

- Expert Advisor scripts exist as discrete Python disk files that can be assigned and run on Chartboard views as required.
- Disk based collections of scrips can be mapped directly into a Chartboard docked explorer via the Expert Advisor mapping facilities.
- Python extension or callback functions (CBEF) facilitate access to the internal Chartboard data structures.
- Python Chartboard extension classes (CBEC) facilitate structured access via the CBEF's.
- Integrated Chartboard debug console for developing and testing Expert-Advisor scripts.
- Period cursor access with stepping through the underlying data series.
- Chart shading, drawing and annotation facilities.
- Trade modelling with dynamic position statements and summaries via properties grid.
- Python integration with subsequent 3rd Party products, Machine Learning, Neural Nets and SQL etc.

Expert advisor scripts exist as disk files and may be exchanged independently of Chartboard. Multiple scripts may be active across multiple views.

CRoot extensions & base class

Python root access to the Chartboard internals is facilitated through the following extension functions.

- Chartboard version and build numbers
  str (=nn.nn.nn and **nnnnn)**
  P2Root.Version() and **Build()**
- Check if stack exists, 'this' refers to stack with focus
  bool
  P2Root.StackExists(sStack)
- Fetch stack type
  str (='OHLC' or 'PanF')
  P2Root.StackType(sStack)
- Open chart stack and return handle
  int (=handle aka **hStack)**
  P2Root.StackOpen(sStack['this'])

Variables persist between instances of a script but discarded when script discontinued or swapped.

- Instance number
  int (=zero based increments after each pass**)**
  P2Root.Instance()
- Declare integer(i) and double(d) variables
  int or **dble**
  P2Series.Declvar_*(sName,*Value,bReset)
- Set integer(i) and double(d) values
  int or **dble**
  P2Series.Setvar_*(sName,*Value)
- Get integer(i) and double(d) values
  int or **dble**
  P2Series.Getvar_*(sName,*Value)

The **CBEC** wraps above functions into the C**Root** singleton class that acts as factory and parent for all subsequent **CStack** derived classes.

Class CRoot
__unit_ (self):
StackExists(self,sStack) → bool:
StackType(self,sStack) → str: 'OHLC' or 'PanF'
StackFactory(self,sStack) → CStackOHLC|PanF:
Version(self) → str:
Build(self) → str:
Instance(self) → int: (0, 1, 2, …)
Declvar_i|d(self, sName, value, bReset) → int | double:
Setvar_i|d(self, sName, value) → int | double:
Getvar_i|d(self, sName, value) → int | double:

CStack extensions & base class

Views containing Chart stacks are internally managed through **Stack** objects that are subsequently exposed to the python environment via the following extension functions

- Confirm or create temporary chart in stack
  str (='exists'. 'enabled', 'created'
  P2Stack.Prerequisites(hStack,sChart)
- Check if chart exists in Stack
  str (='exists'. 'disabled', 'nochart'
  P2Stack.ChartExists(hStack,sChart)
- Rewind chart stack cursors.
  bool
  P2Stack.Rewind(hStack)
- Step chart stack cursor backwards or forwards
  bool
  P2Stack.Step(hStack,ePUnits,nBoF)

- Set shade bar value for all stack charts
  bool
  P2Stack.SetSBar(hStack,ePUnits,nBoF.eSBType,iValue)
- Open chart and return handle
  int (=handle aka **hChart)**
  P2Stack.ChartOpen(sChart])
- Create chart and return handle
  int (=handle aka **hChart)**
  P2Stack.ChartCreate(sChart])
- Declare integer and double variables
  int or **dble**
  P2Stack.Declvar_i|d(hStack,sName,val,bReset[=0])
- Get integer and double variables
  int or **dble**
  P2Stack.Getvar_i|d(hStack,sName)
- Set integer and double variables
  int or **dble**
  P2Stack.Setvar_i|d(hStack,sName,value)

CBEC wraps above functions into the **CStack** base class. Collections of **CStack** objects are managed under the **CRoot** object.

Class CStack
__unit_ (self,sStack[='this']):
ChartExists(self,sChart) → str:
ChartType(self,sChart) → str: 'OHLC', 'RSI', etc
ChartFactory(self,sChart) → ChartOHLC|RSI|etc:
Decl_i|d(self,sName,value,rst[=0]) → int|dble:
Get_i|d(self,sName) → int|dble:
Set_i|d(self,sName,value) → int|dble:
StockCode(self) → str:

*Collections of **CStack** derived objects are managed under the **CRoot** object.*

CStackOHLC – OHLC Chart Stack

Class CStackOHLC
__unit_ (hStack,sChart):
ChartFactory(self,sSeries) → ChartAroon:

***ChartAroon** objects are managed under **CStackOHLC** objects.*

CStackPFigure – Point & Figure Chart Stack

Class CStackFPigure
__unit_ (hStack,sChart):
ChartFactory(self,sSeries) → ChartAroon:

***ChartAroon** objects are managed under **CStackOHLC** objects.*

Chart extensions & base class

Charts within chart stacks internally managed through Chart objects that are subsequently exposed to the python environment via the following extension functions

- Confirm or create temporary **DSeries** on **Chart**
  str (='exists'. 'enabled', 'created'
  P2Chart.Prerequisites(hChart,sDSeries)
- Check if DSeries exists within Chart
  str (='exists'. 'disabled', 'noseries'
  P2Chart.SeriesExists(hStack,sDSeries).
- Fetch DSeries type
  str ('OHLC', 'BB', 'RSI', 'PBV', etc)
  P2Chart.SeriesType(sStack)
- Select configured (i) and (d) parameters
  int or **dble**
  P2Chart.GetParam_*(hChart,sParam,ePUnits,nOFset)
- Select calculated (i) and (d) values
  int or **dble**
  P2Chart.GetValue_*(hChart,sValue,ePUnits,nOFset)
- Open DSeries and return handle.
  int (=handle aka hSeries)
  P2Series.SeriesOpen(hChart,sSeries)

The **CBEC** wraps above functions into the **Chart** base class that is common across all **Chart** derived classes.

Class Chart
__unit_ (self,hStack,sChart):
SeriesExists(self,sSeries) → str:
SeriesType(self,sDSeries) → str: assigned acronym
GetValue_i(self,sName,ePUnits,nOFset) → int:
GetValue_d(self,sName,ePUnits,nOFset) → dble:
GetParam_i(self,sName) → int:
GetParam_d(self,sName) → dble:

*Collections of **Chart** derived objects are managed under **CStack** objects.*

ChartAroon – Aroon

Class ChartAroon
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesAroon:

***ChartAroon** objects are managed under **CStackOHLC** objects.*

ChartADX – Average Direction Index (ADX)

Class ChartADX
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesADX:

***ChartADX** objects are managed under **CStackOHLC** objects.*

ChartATR – Average True Range (ATR)

Class ChartATR
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesATR:

***ChartATR** objects are managed under **CStackOHLC** objects.*

ChartChaikin and ChartCMF

Class ChartChaikin
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesChaikin:

Class CMF
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesCMF:

ChartChaikin and ChartCMF objects are managed under CStackOHLC objects.

ChartCoppock

Class ChartCoppock
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesCoppock:

ChartCoppock objects are managed under CStackOHLC objects.

ChartCCI – Commodity Channel Index

Class ChartCCI
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesCCI:

***ChartCCI** objects are managed under **CStackOHLC** objects.*

ChartDPO – Detrended Price Oscillator

Class ChartDPO
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesDPO:

***ChartDPO** objects are managed under **CStackOHLC** objects.*

ChartEFI – Elder Ray

Class ChartEFI
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesEFI:

***ChartEFI** objects are managed under **CStackOHLC** objects.*

ChartKST – Pring's Know Sure Thing

Class ChartKST
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesKST:

***ChartKST** objects are managed under **CStackOHLC** objects.*

ChartMACD – Moving Av. Cumulative Dist.

Class ChartMACD
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesMACD:

***ChartMACD** objects are managed under **CStackOHLC** objects.*

ChartMFI – Money Flow Index

Class ChartMFI
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesMFI:

***ChartMFI** objects are managed under **CStackOHLC** objects.*

ChartOBV – On Balance Volume

Class ChartOBV
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesOBV:

***ChartOBV** objects are managed under **CStackOHLC** objects.*

ChartPMO – Price Momentum Oscillator

Class ChartPMO
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesPMO:

***ChartPMO** objects are managed under **CStackOHLC** objects.*

ChartPVO – Percentage Volume Oscillator

Class ChartPVO
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesPVO:

***ChartPVO** objects are managed under **CStackOHLC** objects.*

ChartROC – Rate of Change

Class ChartROC
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesROC:

***ChartROC** objects are managed under **CStackOHLC** objects.*

ChartRSI – Relative Strength Index

Class ChartRSI
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesRSI:

***ChartRSI** objects are managed under **CStackOHLC** objects.*

ChartStochRSI – Stochastics RSI

Class ChartStochRSI
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesStochRSI:

***ChartStochRSI** objects are managed under **CStackOHLC** objects.*

ChartSTO – Stochastics Fast, Slow and Full

Class ChartSTO
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesSTO:

***ChartSTO** objects are managed under **CStackOHLC** objects.*

ChartTRIX – Triple smoothed Exp Moving Av.

Class ChartTRIX
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesTRIX:

***ChartTRIX** objects are managed under **CStackOHLC** objects.*

ChartTSI – True Strength Index

Class ChartTSI
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesTSI:

***ChartTSI** objects are managed under **CStackOHLC** objects.*

ChartSTDEV – Volatility or Standard Deviation

Class ChartSTDEV
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesSTDEV:

***ChartSTDEV** objects are managed under **CStackOHLC** objects.*

ChartVolume – Trade Volume

Class ChartVolume
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesVolume:

***ChartVolume** objects are managed under **CStackOHLC** objects.*

ChartShorts- Stock Shorts

Class ChartShorts
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesShorts:

***ChartShorts** objects are managed under **CStackOHLC** objects.*

ChartVTX – Vortex Indicator

Class ChartVTX
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesVTX:

***ChartVTX** objects are managed under **CStackOHLC** objects.*

ChartWmR – Williams %R

Class ChartWmR
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesWmR:

***ChartWmR** objects are managed under **CStackOHLC** objects.*

ChartZigZag

Class ChartZigZag
__unit_ (hStack,sChart):
DSeriesFactory(self,sSeries) → DSeriesZigZag:

ChartZigZag *objects are managed under* ***CStackOHLC*** *objects.*

DSeries extensions & base class

Chartboard data sets and calculations are internally managed through DSeries objects that are subsequently exposed to the python environment via the following CBEF's

- Get configured (i) and (d) parameters
  int or **dble**
  P2Series.GetParam_*(hSeries,sParam)
- Set (i) and (d) parameters
  int or **dble**
  P2Series.SetParam_*(hSeries,sParam,i|dValue)
- Select calculated (i) and (d) values
  int or **dble**
  P2Series.GetValue_*(hDSeries,sValue,ePUnits,nOSet)
- Set EA shade bar
  int (updated shade bars)
  P2Series.EA_SetShadeBar(hSeries
  ,ePUnits,nOSet,nSBType,iValue)
- Get EA shade bar
  int (selected shade bar value)
  P2Series.EA_GetShadeBar(hSeries
  ,ePUnits,nOSet,nSBType)
- Clear EA shade bar(s)
  int (cleared shade bars)
  P2Series.EA_ClearShadeBars(hSeries,ePUnits)
  ePUnits = 0 for all period units.

The CBEC's wrap above functions into the **DSeries** base class that is common across all **Chart** classes.

Class DSeries
__unit_ (self,hChart,sDSeries):
Exists(self) → bool:
GetValue_d(self,sValue,ePUnits,nOFset) → dble:
GetValue_i(self,sValue,ePUnits,nOFset) → int:
GetParam_i(self,sParam) → int:
GetParam_d(self,sParam) → dble:
#Shade Bars
EA_GetShadeBar(self,ePUnits,nOSet) → dble:
EA_SetShadeBar(self,ePUnits,nOSet,iState) → int:
EA_ClearShadeBar(self,ePUnits) → int:

Collections of ***DSeries*** *derived objects are managed under the* ***Chart*** *objects.*

DSeriesBB – Bollinger Bands

Class DSeriesBB
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'BB+' Upper band
'BBsma' Simple Moving Average
'BB' Lower Band
GetValue_i(…) → int:
GetParam_i(…) → int:
'Kvalue'
'SMAperiods'
GetParam_d(…) → dble:

BB *data series accessible from* ***ChartOHLC*** *objects.*

DSeriesChandelier – Chandelier Exit

OHLC Chart Chandelier Exit overlay

DSeriesIchimoku – Ichimoku Cloud

Class DSeriesIchimoku
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'TenkouSen' value
'KijunSen' value
'ChikouSen' value
'SenkouASen' value
'SenkouBSen' value
GetValue_i(…) → int:
GetParam_i(…) → int:
'Tenkanperiods'
'Kijunperiods'
'Senkouperiods'
'Chikouperiods'
GetParam_d(…) → dble:

Ichimoku *data series accessible from* ***ChartOHLC*** *objects.*

DSeriesKAMA – Kaufman's Adaptive Moving Av

Class DSeriesKAMA
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'KAMA' Calculated value
GetValue_i(…) → int:
GetParam_i(,sParam) → int:
'ERperiods'
'FASTperiods'
'SLOWperiods'
GetParam_d(…) → dble:

KAMA *data series accessible from* ***ChartOHLC*** *objects.*

DSeriesKeltner – Keltner Channels

Class DSeriesKeltner
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'KeltnerHi' Calculated high value
'KeltnerLo' Calculated low value
GetValue_i(…) → int:
GetParam_i(…) → int:
'EMAperiods'
'ATRperiods'
GetParam_d(…) → dble:

Keltner *data series accessible from* ***ChartOHLC*** *objects.*

DSeriesSMA – Simple Moving Average

OHLC Chart Simple Moving Average (SMA) overlay

DSeriesEMA – Exponential Moving Average

OHLC Chart Exponential Moving Average (EMA) overlay

DSeriesSAR – Parabolic Set and Retrace

Class DSeriesSAR
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'SAR' Caculated SAR value
GetValue_i(…) → int:
'AoB' Above(+1) or Below(-1) flag
'AoBage' Above or below age in period units
GetParam_i(…) → int:
GetParam_d(…) → dble:
'AF' Acceleration factor
'AFmax'. Acceleration factor max.

SAR *data series accessible from* ***ChartOHLC*** *objects.*

DSeriesDonchian – Price or Donchian Channels

Class DSeriesDonchian
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'DChi' Calculated high channel value

'DC' Calculated price channel value
'DClo' Calculated low channel value
GetValue_i(…) → int:
GetParam_i(…) → int:
'PCperiods'
GetParam_d(…) → dble:

Donchian *data series accessible from* ***ChartOHLC*** *objects.*

DSeriesReversals – Candlestick Patterns

Class DSeriesReversals
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
GetValue_i(…) → int:
'All' Reversal type
GetParam_i(sParm) → int:
'RTYPE-Mask'
GetParam_d(…) → dble:
SetParam_i(sParm,iValue) → int:
'RTYPE-Mask'

PChan *data series accessible from* ***ChartOHLC*** *objects.*

DSeriesMACD – Moving Average Cumulative Dist

Class DSeriesMACD
__unit_ (hChart, sSeries):
GetValue_d(,sValue,,) → dble:
'MACD' Calculated value
'MACDsignal' Calculated signal line value
'MACDiff' Difference
GetValue_i(,sValue,,) → int:
'BoS' Buy(+1) or sell(-1) flag
'BoSage' Buy or sell age in period units
GetParam_i(,sParm,,) → int:
'EMA1periods'
'EMA2periods'
'Signalperiods'
GetParam_d(…) → dble:

MACD *data series accessible from* ***ChartMACD objects.***

DSeriesRSI – Relative Strength Indicator

Class DSeriesRSI
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'RSI' Calculated value
GetValue_i(…) → int:
GetParam_i(,sParm,,) → int:
'RSIperiods'
GetParam_d(,sParm,,) → dble:
'OBought' Over bought line
'OSold' Oversold line

RSI *data series accessible from* ***ChartRSI*** *objects.*

DSeriesTSI – True Strength Indicator

Class DSeriesTSI
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'TSI' Calculated value
'TSIsignal' Signal line
GetValue_i(,sValue) → int:
'BoS' Buy(+1) or sell(-1) flag
'BoSage' Buy or sell age in period units
GetParam_i(,sParm,,) → int:
'PC1periods'
'PC2periods'
'Signalperiods'
GetParam_d(…) → dble:

TSI *data series accessible from* ***ChartTSI*** *objects.*

DSeriesAroon – Aroon Indicator

Class DSeriesAroon
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'AroonHi' Calculated period high
'AroonLo' Calculated period low
'AroonDiff' Difference
GetValue_i(,sValue) → int:
'BoS' Buy(+1) or sell(-1) flag
'BoSage' Buy or sell age in period units
GetParam_i(,sParm,,) → int:
'Aroonperiods'
GetParam_d(…) → dble:

Aroon *data series accessible from* ***ChartAroon*** *objects.*

DSeriesADX – Average Directional Index

Class DSeriesADX
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'ADX' Calculated value
'DI+' Directional Index plus
'DI-' Directional Index minus
GetValue_i(…) → int:
GetParam_i(,sParm,,) → int:
'ADXperiods'
GetParam_d(…) → dble:

ADX *data series accessible from* ***ChartADX*** *objects.*

DSeriesATR – Average True Range

Class DSeriesATR
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'ATR' Calculated value
'Hi' High value
'Lo' Low value
'Difference' Between Hi and Lo
GetValue_i(…) → int:
GetParam_i(,sParm,,) → int:
'ATRperiods'
GetParam_d(…) → dble:

ATR *data series accessible from* ***ChartATR*** *objects.*

DSeriesChaikin – Chaikin Indicator

Class DSeriesChaikin
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'Chaikin' Calculated value
GetValue_i(…) → int:
GetParam_i(,sParm,,) → int:
'FASTperiods' Fast periods
'SLOWperiods' Slow periods
GetParam_d(…) → dble:

Chaikin *data series accessible from* ***ChartChaikin*** *objects.*

DSeriesCMF – Chaikin Money Flow

Class DSeriesCMF
__unit_ (hChart, sSeries):
GetValue_d(,sValue,,) → dble:
'CMF' Calculated value
GetValue_i(…) → int:
GetParam_i(…) → int:
'CMFperiods'
GetParam_d(…) → dble:

CMF *data series accessible from* ***ChartCMF*** *objects.*

DSeriesCoppock – Coppock Indicator

Class DSeriesCoppock
__unit_ (hChart, sSeries):

GetValue_d(,sValue,,) → dble:
'Coppock' Calculated value
GetValue_i(…) → int:
GetParam_i(,sParm,,) → int:
'ROCAperiods'
'ROCBperiods'
'WMAperiods'
GetParam_d(…) → dble:

Coppock *data series accessible from* ***ChartCoppock*** *objects.*

DSeriesCCI – Commodity Channel Index

Class DSeriesCCI
__unit_ (hChart, sSeries):
GetValue_d(,sValue,,) → dble:
'CCI' Calculated value
GetValue_i(…) → int:
GetParam_i(,sParm,,) → int:
'CCIperiods'
GetParam_d(…) → dble:

CCI *data series accessible from* ***ChartCCI*** *objects.*

DSeriesDPO – Detrended Price Oscillator

Class DSeriesDPO
__unit_ (hChart, sSeries):
GetValue_d(,sValue,,) → dble:
'DPO' Calculated value
GetValue_i(…) → int:
GetParam_i(…) → int:
'DPOperiods'
GetParam_d(…) → dble:

DPO *data series accessible from* ***ChartDPO*** *objects.*

DSeriesEFI – Elder Ray

Class DSeriesEFI
__unit_ (hChart, sSeries):
GetValue_d(,sValue,,) → dble:
'EFI' Calculated value
GetValue_i(…) → int:
GetParam_i(,sParm) → int:
'EFIperiods'
GetParam_d(…) → dble:

EFI *data series accessible from* ***ChartEFI*** *objects.*

DSeriesKST – Prings Know Sure Thing

Class DSeriesKST
__unit_ (hChart, sSeries):
GetValue_d(,sValue,,) → dble:
'KST' Calculated value
'KSTsignal' Signal line
GetValue_i(,sValue) → int:
'BoS' Buy(+1) or sell(-1) flag
'BoSage' Buy or sell age in period units
GetParam_i(…) → int:
GetParam_d(…) → dble:

KST *data series accessible from* ***ChartKST*** *objects.*

DSeriesMFI – Money Flow Index

Class DSeriesMFI
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'MFI' Calculated value
GetValue_i(…) → int:
GetParam_i(,sParm,,) → int:
'MFIperiods'
'OBought' Over bought line
'OSold' Over sold line
GetParam_d(…) → dble:

MFI *data series accessible from* ***ChartMFI*** *objects.*

DSeriesOBV – On Balance Volume

Class DSeriesOBV
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'OBV' Calculated value
GetValue_i(…) → int:
GetParam_i(…) → int:
GetParam_d(…) → dble:

OBV *data series accessible from* ***ChartOBV*** *objects.*

DSeriesPMO – Price Momentum Oscillator

Class DSeriesPMO
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'PMO' Calculated value
'PMOema' Exponential Moving Average
GetValue_i(…) → int:
GetParam_i(,sParm,,) → int:
'PMO1periods'
'PMO2periods'
'EMAperiods'
GetParam_d(…) → dble:

PMO *data series accessible from* ***ChartPMO*** *objects.*

DSeriesPVO – Price Volume Oscillator

Class DSeriesPVO
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
GetValue_i(,sValue) → int:
'BoS' Buy(+1) or sell(-1) flag
'BoSage' Buy or sell age in period units
GetParam_i(…) → int:
GetParam_d(…) → dble:

PVO *data series accessible from* ***ChartPVO*** *objects.*

DSeriesROC – Rate of Change

Class DSeriesROC
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'ROC' Calculated value
GetValue_i(…) → int:
GetParam_i(,sParm,,) → int:
'ROCperiods'
GetParam_d(…) → dble:

ROC *data series accessible from* ***ChartROC*** *objects.*

DSeriesStochRSI – Stochastics RSI

Class DSeriesStockRSI
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
GetValue_i(…) → int:
GetParam_i(…) → int:
GetParam_d(…) → dble:

StockRSI *data series accessible from* ***ChartStochRSI*** *objects.*

DSeriesSTO – Stochastics Fast, Slow & Full

Class DSeriesSTO
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'FastK'
'FastD'

'SlowK'
'SlowD'
'FullK'
'FullD'

GetValue_i(…) → int:
GetParam_i(,sParm,,) → int:
'Kperiods'
'Dperiods'
'Xperiods'
GetParam_d(…) → dble:

STO *data series accessible from* ***ChartSTO*** *objects.*

DSeriesTRIX – Triple Smoother Exp. Moving Avg.

Class DSeriesTRIX
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'TRIX' Calculated value
'TRIXsignal' Signal line
GetValue_i(,sValue) → int:
'BoS' Buy(+1) or sell(-1) flag
'BoSage' Buy or sell age in period units
GetParam_i(…) → int:
'EMAperiods'
'Signalperiods'
GetParam_d(…) → dble:

TRIX *data series accessible from* ***ChartTRIX*** *objects.*

DSeriesSTDEV – Standard Deviation or Volatility

Class DSeriesSTDEV
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'STDEV' Calculated value
GetValue_i(…) → int:
GetParam_i(,sParm,,) → int:
'STDEVperiods'
GetParam_d(…) → dble:

STDEV *data series accessible from* ***ChartSTDEV*** *objects.*

DSeriesShorts – Short Sales Volume

Class DSeriesShorts
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'Shorts' Total shorts
'PoT' Percentage of Total
'Delta' Difference over period
GetValue_i(…) → int:
GetParam_i(…) → int:
GetParam_d(…) → dble:

Shorts *data series accessible from* ***ChartOHLC*** *and* ***ChartShorts*** *objects.*

DSeriesVolume – Trade Volume

Class DSeriesVolume
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'Volume' Total volume for period
GetValue_i(…) → int:
GetParam_i(,sParm) → int:
GetParam_d(…) → dble:

Volume *data series accessible from* ***ChartOHLC*** *and* ***ChartVolume*** *objects.*

DSeriesVTX – Vortex Indicator

Class DSeriesVTX
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'VM+' Positive trend movement
'VM-' Negative trend movement
GetValue_i(…) → int:
GetParam_i(,sParm) → int:
'VTXperiods'
GetParam_d(…) → dble:

VTX *data series accessible from* ***ChartVTX*** *objects.*

DSeriesWmR – Williams Indicator

Class DSeriesWmR
__unit_ (hChart,sSeries):
GetValue_d(,sValue,,) → dble:
'WmR' Calculated value
GetValue_i(…) → int:
GetParam_i(,sParm) → int:
'WmRperiods'
GetParam_d(…) → dble:

WmR *data series accessible from* ***ChartWmR*** *objects.*

DSeriesZigZag – Zig-Zag Line

Class DSeriesZigZag
__unit_ (hChart, sSeries):
GetValue_d(,sValue,,) → dble:
'ZigZag' Calculated value
GetValue_i(…) → int:
GetParam_i(…) → int:
GetParam_d(…) → dble:
'ZZpercent' Minimum variance

ZigZag *data series accessible from* ***ChartOHLC*** *objects.*

Trend extensions & base class

Chartboard supports both the access to existing Trend objects and annotation of new objects on Charts whilst Expert Advisor scripts are running.

-
-

The **CBEC** wraps above functions into the **DTool** base class that is common across all **DTool** derived classes.

Class DTool
__unit_ (hSeries):

Collections of ***DTool*** *derived objects are managed under* ***Chart*** *objects.*

Drawing extensions & base class

Chartboard supports drawing objects for annotations on Charts whilst Expert Advisor scripts are running.

- Perform and annotate model buy trade on Chart
  int (=assigned TagID)
  P2Draw.AttachTag(hChart,ePUnits,sTag,sMsg).
- Perform and annotate model sell trade on Chart
  int (=modelled trade Id)
  P2Draw.RemoveTag(hChart,nTagID)

The **CBEC** wraps above functions into the **Draw** base class that is common across all **Draw** derived classes.

Class Draw
__unit_ (hChart,hSeries):

Collections of ***Draw*** *derived objects are managed under* ***Chart*** *objects.*

DrawTag – Draw Chart Tags

Class DrawTag
__unit_ (hChart,sSeries):
DoDrawTag(self, ePUnits, sTag, sMsg) → int:

DrawTag objects are managed under Chart objects.

Modelling extensions & base class

Chartboard supports both trade modelling and annotation through the following extension functions. Modelling status is viewed via the properties grid associated with annotated trades.

- Perform and annotate model buy trade on Chart
  int (=modelled trade Id)
  P2Model.PerformBuy(hChart,sUnits,sTag,sMsg
  ,dValue).
- Perform and annotate model sell trade on Chart
  int (=modelled trade Id)
  P2Model.PerformSell(hChart,sUnits,sTag,sMsg
  ,nTrdID)

The **CBEC** wraps above functions into the **ModelTrade** base class that is common across all **ModelTrade** derived classes.

Class ModelTrade
__unit_ (self,hChart):
PerformBuy(self.hChart,e) → int: assigned Trade Id
PerformSell(self,hChart,ePUnits,sTag,sMsg) → int:
Quantity(self,dQuantity) → dble:
self.dQuantity = dQuantity
Value(self,dValue) → int:
self.dValue = dValue

*Collections of **ModelTrade** derived objects are managed under **Chart** objects.*