NinjaScript Code-Breaking Changes for NinjaTrader 7

Revision 11, October 11, 2010

General

- Data type for volume changed from `int` to `long` for the following methods and properties

```
GetCurrentAskVolume()
GetCurrentBidVolume()
MarketDataEventArgs.Volume
MarketDepthEventArgs.Volume
```

- `TickSize` is no longer reflective of splits. For splits use this:

```
double tickSizeSplitAdjusted = TickSize /
Bars.Instrument.MasterInstrument.Splits.GetSplitFactor(Bars.GetSessionDate(Ti
me[0]));
```

- `Bars.GetSessionBar.Time` now reflects the session end time's timestamp instead of 12:00 AM
- `BarsPeriod.Id` and `BarsPeriod.Value` should not be used without additional logic in the `Add()` method. Logic should be placed to filter the period type to ensure compatibility with new period types. Please use the following:

```
AddKagi(string instrumentName, PeriodType basePeriodType, int
basePeriodTypeValue, int reversal, ReversalType reversalType, MarketDataType
marketDataType);

AddRenko(string instrumentName, int brickSize, MarketDataType
marketDataType);

AddPointAndFigure(string instrumentName, PeriodType basePeriodType, int
basePeriodTypeValue, int boxSize, int reversal, PointAndFigurePriceType
pointAndFigurePriceType, MarketDataType marketDataType);

AddLineBreak(string instrumentName, PeriodType basePeriodType, int
basePeriodTypeValue, int lineBreakCount, MarketDataType marketDataType);
```

- `CurrentBar` internal pointers are now updated earlier. This ensures that accessing bars across multiple series will be in sync regardless of which `BarsInProgress` context you may be working out of.
  - For historical, pointers for all bar series with the same timestamp will be updated before `OnBarUpdate()` will be triggered.
  - For real-time, pointers for all bar series of the same instrument will be updated before `OnBarUpdate()` will be triggered.
- User parameters are no longer marked by `[Category("Parameters")]`. Please use this:

```
[GridCategory("<AnyText>")]
```

  Note: You should not use category names that are the same as any internal NinjaTrader categories besides the default "Parameters" category.

- `Bars.SessionBegin` and `Bars.SessionEnd` are deprecated. Please use this:

```
BarsArray[int barSeries].Session.GetNextBeginEnd(DateTime time, out DateTime
sessionBegin, out DateTime sessionEnd)
```

- `BarColorSeries` should not be used. Please use this:

```
ColorSeries[int barsAgo]
```

- `BarColor` no longer colors the entire bar. It will color the bar body only. To color the bar outline please use `CandleOutlineColor`.
- DataSeries objects now default to storing only the last 256 data points with `MaximumBarsLookBack.TwoHundredFiftySix`. If you need more stored points please use this:

```
myDataSeries = new DataSeries(this, MaximumBarsLookBack.Infinite);
```

- `Pen` objects now have a minimum width of 1
- Some previously undocumented draw method signatures have been removed. Please see the Help Guide article of your particular draw method for a list of acceptable signatures that need to be used.
- `TriggerCustomEvent()` signature has changed. Please use this:

```
TriggerCustomEvent(CustomEvent customEvent, int barsInProgress, object state)
```

- When working with custom DataSeries objects, do NOT access a DataSeries value on a bar where you have not called `DataSeries.Set()` since you can run into exceptions
    - It is advisable to always call `.Set()` and use a default value you can filter for in your code even on the bars you do not wish to set a value for
- `ChartControl.ChartStyle.Pen` cannot be set. Please instead use the property you are interested in directly:

```
ChartControl.ChartStyle.Pen.Color = Color.Transparent;
```

- `Bars.GetSessionDate()` is no longer publicly exposed. Please use this instead:

```
Bars.GetTradingDayFromLocal(DateTime time)
```

- `Bars.GetSessionBar()` is now deprecated. Please use this instead:

```
Bars.GetDayBar(int sessionsAgo)
```

NinjaScript Indicators

- Wrapper generator has been changed. NinjaScript archives need to be regenerated with the new wrappers.
    - Open the indicator in the NinjaScript Editor, recompile, then re-export the NinjaScript
    - If your distribution method was through NinjaTrader backups you will need to recreate the backup files with NinjaTrader 7 to get the new wrappers for the indicators
- "`//`" in string literals will work now. Example:

```
[Description("Chris Carolan's Net-Lines reversal pattern described at
http://carolan.org/indicators/")]
```

- If overriding the Plot method, you will need to accommodate for the possibility of running on the new non-equidistant charts*. If these changes are not made, you may run into exceptions on non-equidistant charts.

    - For examples and best practices, please see either the Pivots, RegressionChannel, or ZigZag system indicators
    - Iterating through bars should now be done with this code:

```
for (int idx = this.LastBarIndexPainted; idx >=
Math.Max(this.FirstBarIndexPainted, this.LastBarIndexPainted); idx--)
```

    - x/y positions should now be accessed by this code:

```
int x = ChartControl.GetXByBarIdx(BarsArray[0], idx);
int y = ChartControl.GetYByValue(this, val);
```

* Non-equidistant charts do not have fixed distances between each bar like they do in traditional (equidistant) charts. Instead they will have a fixed x-axis timeline where every inch along the axis represents the same amount of time. Benefits of this chart type include being able to gauge momentum on non-time based charts like ticks or volume by visualizing how long it takes to finish building the next bar.

NinjaScript Strategies

- `IExecution.IOrder` property will now reflect the order state at the time of `OnExecution()` meaning it could return values ahead of the currently being processed `IExecution`.
    - Ex: There can be discrepancies between `IExecution.Order.Filled` and `IOrder.Filled` as accessed during order placement
- `IOrder` objects are now unique throughout the lifetime of a strategy
    - Do *NOT* hold onto `.Token` values since they will change as the strategy goes from historical to live
    - To check for equality you can directly compare `IOrders`. Example:

```
if (entryOrder == order)
```

    - `IOrder.Action` has been renamed to `IOrder.OrderAction`
- All pending orders are processed after `OnBarUpdate()` now. Before only orders targeting that particular `BarsInProgress` would process
- Manual exit orders `(ExitLong()/ExitLongLimit()/etc)` no longer automatically amend quantity as new fills add to the existing position
- `OnPositionUpdate()` will now only trigger when a position has changed
- `BarsRequired` is skipped in the rolled forward period of a walk forward optimization
- `BarsSinceEntry()` returns proper values in relation to multi-series strategies and multiple entry signals. If you used this before you may need to adjust your logic.
- `BarsSinceExit()` now reports -1 instead of erroneous reports of 0. Please revise your logic to reflect this.
- When working with indicators in a strategy, if you run into exceptions or returns of unexpected values, make sure you have followed the new `.Set()` requirements discussed above in *General*. Also ensure that `Update()` is called in any method within your indicator that returns values.