|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) > [Educational Resources](https://ninjatrader.com/support/helpGuides/nt8/educational_resources.htm) >  **Multi-Time Frame & Instruments** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/multi-threading.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/educational_resources.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/understanding_the_lifecycle_of.htm) |

**Multi-Series Scripting Overview**

NinjaScript supports multiple time frames and instruments in a single script. This is possible because you can add additional Bars objects to indicators or strategies, in addition to the primary Bars object to which they are applied. A Bars object represents all of the bars of data on a chart. For example, if you had a MSFT 1 minute chart with 200 bars on it, the 200 bars represent one Bars object. In addition to adding Bars objects for reference or for use with indicator methods, you can execute trades across all the different instruments in a script. There is extreme flexibility in the NinjaScript model that NinjaTrader uses for multiple-bars scripts, so it is very important that you understand how it all works before you incorporate additional Bars objects in a script. An important fact to understand is that NinjaScript is truly event driven; every Bars object in a script will call the [OnBarUpdate()](https://ninjatrader.com/support/helpGuides/nt8/onbarupdate.htm) method. The significance of this will become evident throughout this page.

|  |
| --- |
| **Note**:  If using [OnMarketData()](https://ninjatrader.com/support/helpGuides/nt8/onmarketdata.htm), a subscription will be created on all bars series added in your indicator or strategy strategy (even if the instrument is the same).  The market data subscription behavior occurs both in real-time and during [TickReplay](https://ninjatrader.com/support/helpGuides/nt8/developing_for__tick_replay.htm) historical |

It is also important that you understand the following method and properties:

•[AddDataSeries()](https://ninjatrader.com/support/helpGuides/nt8/adddataseries.htm)

•[BarsArray](https://ninjatrader.com/support/helpGuides/nt8/barsarray.htm)

•[BarsInProgress](https://ninjatrader.com/support/helpGuides/nt8/barsinprogress.htm)

•[CurrentBars](https://ninjatrader.com/support/helpGuides/nt8/currentbars.htm)

|  |
| --- |
| **Note**:  As we move through this section, the term "Primary Bars" will be used and for the purpose of clarification, this will always refer to the first Bars object loaded into a script. For example, if you apply a script on MSFT 1 minute chart, the primary Bars would be MSFT 1 minute data set.    **This section is written in sequential fashion. Example code is re-used and built upon from sub section to sub section.** |

tog_minus        [Working With Multi-Time Frame Objects](javascript:HMToggle('toggle','WorkingWithMultiTimeFrameObjects','WorkingWithMultiTimeFrameObjects_ICON'))

|  |  |  |
| --- | --- | --- |
| **Data processing sequence**  Understanding the sequence in which bars series process and the granularity provided by market data vendors is essential for efficient multi-series development. Let’s assume we have two series (primary and secondary) in our script, which is representing the same instrument, yet different intervals.  During historical data processing, NinjaTrader updates the two series *strictly* according to their timestamps, calling the primary bar series of the corresponding timestamps first, and then calling the secondary series.     |  | | --- | | **Note**:  Historical bars are processed according to their timestamps with the primary bars first, followed by the secondary, which is **NOT** guaranteed to be the same sequence that these events occurred in real-time.  If your development requires ticks to process in the same sequence historically as well as in real-time, you will need to enable [Tick Replay](https://ninjatrader.com/support/helpGuides/nt8/developing_for__tick_replay.htm) (utilizes more PC resources). |       **Shared Timestamps**  In circumstances where multiple bars share the same exact timestamps, your primary bars series will *always* be processed first, followed by the secondary bars series (regardless of the period value used). Consequently, if you were looking to obtain a value from the secondary bars series, it would **ONLY** be available *after* the primary series has been processed for the same timestamps. For example, consider a news event or a fast moving market with an influx of ticks (session begin/session end). This activity will yield a wider range of bars than usual and the probability of those bars sharing the same timestamps increases. If such a succession of bars with the same timestamps is processed, the primary bars would be processed first and then the secondary bars during this period.     |  | | --- | | **Tip**: While the following behavior applies to all period types, the effects are amplified on smaller time frames.  If you plan on using a high-resolution (e.g., 1-second, 10-tick, etc), please make sure to thoroughly read and understand the material below when working with these additional series.  It is also important to keep in mind that the granularity of the timestamps will dictate how accurately NinjaTrader can synchronize the bars in historical processing.   The available level of granularity will be dependent upon which [data provider](https://ninjatrader.com/support/helpGuides/nt8/data_by_provider.htm) you use with NinjaTrader. |       Let’s look at an illustration of how the multi-time frame bar processing sequence can be understood.  Assume our primary series is a 5-tick bar series, and our secondary series is a 1-tick bar series.  The time of day is near the session close, so a rapid sequence of bars is generated.    In the figure below, the 1st group of bars (colored orange), and the 4th group of bars (colored purple) process in an exact logical sequence (i.e., a single primary bar update, followed by five secondary series updates).  This is because each bar in these groups have *unique timestamps* and NinjaTrader can synchronize those bars logically in the exact time sequence each series updated.  However, all of the bars marked with red text share the*same exact timestamps* down to the millisecond (14:59:00:480).  Since there were six ticks in sequence with the shared timestamps, this range of ticks expands two of the primary bars (colored green and blue).  As a result, the primary bar #3 appears to update earlier when compared to the secondary series.  In reality, both bars series are incrementing in their exact sequence according to the timestamps of each series.      same_timestamp_bars  Figure 1.  Bar processing  with shared timestamps  1.Timestamps of primary series (hour, minute, second, millisecond)  2.Current bars numbered in series representing 5-tick primary series  3.Current bars numbered in series representing 1-tick secondary series  4.Millisecond time stamps of secondary series  5.A sequence of bars sharing the same time stamps |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?multi-time_frame__instruments.htm#WorkingWithMultiTimeFrameObjects)

tog_minus        [Adding Additional Bars Objects to NinjaScript](javascript:HMToggle('toggle','AddingAdditionalBarsObjectToninjascript','AddingAdditionalBarsObjectToninjascript_ICON'))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Additional Bars are added to a script via the [AddDataSeries()](https://ninjatrader.com/support/helpGuides/nt8/adddataseries.htm) method in the [OnStateChange()](https://ninjatrader.com/support/helpGuides/nt8/onstatechange.htm) method when the [State](https://ninjatrader.com/support/helpGuides/nt8/state.htm) has reached **State.Configure**. When a Bars object is added to a script, it is also added to the [BarsArray](https://ninjatrader.com/support/helpGuides/nt8/barsarray.htm) array. **BarsArray** functions like a container in the script that holds all Bars objects added to the script. As a Bars object is added to the script, it's added to **BarsArray** and given an index number so we can retrieve this Bars object later.     |  | | --- | | **Warning:**  •This method should **ONLY** be called from the [OnStateChange()](https://ninjatrader.com/support/helpGuides/nt8/onstatechange.htm)method during **State.Configure**  •Arguments supplied to **AddDataSeries()** should be hardcoded and **NOT** dependent on run-time variables which cannot be reliably obtained during [State.Configure](https://ninjatrader.com/support/helpGuides/nt8/state.htm) (e.g., [Instrument](https://ninjatrader.com/support/helpGuides/nt8/instrument.htm), [Bars](https://ninjatrader.com/support/helpGuides/nt8/bars.htm), or user input).  Attempting to add a data series dynamically is **NOT** guaranteed and therefore should be avoided.  Trying to load bars dynamically may result in an error similar to: **Unable to load bars series. Your NinjaScript may be trying to use an additional data series dynamically in an unsupported manner.**  •When instantiating indicators in a Multi-Series script in [OnStateChange](https://ninjatrader.com/support/helpGuides/nt8/onstatechange.htm), the input any hosted indicator is running on should be explicitly stated (since a specific [BarsInProgress](https://ninjatrader.com/support/helpGuides/nt8/barsinprogress.htm) is not guaranteed) |     For the purpose of demonstration, let's assume that a MSFT 1 minute bar is our primary Bars object (set when the script is applied to a 1 minute MSFT chart) and that the OnStateChange() method is adding a 3 minute Bars object of MSFT, then adding a 1 minute Bars object of AAPL, for a total of 3 unique Bars objects.     | ns | | --- | | protected override void OnStateChange() {   if (State == State.SetDefaults)   {                     Name   = "Multi-Time Frame & Instruments Example";   }   else if (State == State.Configure)   {       AddDataSeries(BarsPeriodType.Minute, 3);     AddDataSeries("AAPL", BarsPeriodType.Minute, 1);   } } |      |  | | --- | | **Note**: To maximize data loading performance, any NinjaScript object (indicator or strategy as host) which references a multi-series indicator which calls AddDataSeries must include it's own calls to AddDataSeries(). For example, if the code above was included in an indicator, and that indicator was referenced in a  NinjaScript strategy, then the hosting strategy will need to include the same calls to AddDataSeries(). When the strategy adds the additional Bars objects, the calls to AddDataSeries() within the indicator will be ignored. If the Bars objects are not added by the strategy in such cases, and error will be thrown in the Log tab of the Control Center that would read - "A hosted indicator tried to load additional data. All data must first be loaded by the hosting NinjaScript in its configure state." | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?multi-time_frame__instruments.htm#AddingAdditionalBarsObjectToninjascript)

tog_minus        [Creating Series<T> Objects](javascript:HMToggle('toggle','CreatingSeriesObjects','CreatingSeriesObjects_ICON'))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Series<T> Objects**  [Series<T>](https://ninjatrader.com/support/helpGuides/nt8/seriest.htm) is the base class for [PriceSeries](https://ninjatrader.com/support/helpGuides/nt8/priceseries.htm), [TimeSeries](https://ninjatrader.com/support/helpGuides/nt8/timeseries.htm), and [VolumeSeries](https://ninjatrader.com/support/helpGuides/nt8/volumeseries.htm). Rather than using one of these pre-defined derived classes, you can create your own Series<T> collection to hold any Type that you choose. The advantage that Series<T> has over other collections is that it can be quickly initialized to contain a number of index slots equal to the number of bars in one of the Bars objects on the chart, with each index slot corresponding to a specific bar.    **Initializing a Series<T> with BarsArray**  A Series<T> can be constructed by passing in a specific index of BarsArray. Initializing a Series<T> this way produces an empty Series<T> container holding the same number of index slots as the BarsArray that was passed in as an argument. For example, assuming that BarsArray[1] holds 500 bars, the code below will create an empty Series<T> with 500 index slots:     | ns **Initializing Series<T> with BarsArray** | | --- | | private Series<double> myEmptyIndexedSeries; // Define a Series<T> objectvariable.   // Initialize the Series object to have the same number of index slots as BarsArray[1] protected override void OnStateChange() {   if (State == State.DataLoaded)   {       // Passing in BarsArray[1] as an argument results in an empty Series with an identical number of index slots       myEmptyIndexedSeries = new Series<double>(BarsArray[1]);   } } |       This method of initializing a Series<T> can be especially useful when you wish to store user-defined information related to each bar in a Bars object on the chart. This process ensures that index slots are available for every bar on the chart right away.    **Initializing a Series<T> with an Indicator Method**  Passing in an indicator method as an argument when instantiating a Series<T> object provides an alternative to the process outlined above. Because indicator methods already contain Series objects synced to the bars on a chart, they can be used to inform the constructor of Series<T> of how many index slots to create.     | ns **Initializing Series<T> with an Indicator Method** | | --- | | // Declare two Series objects private Series<double> primarySeries; private Series<double> secondarySeries;   protected override void OnStateChange() {     if (State == State.Configure)   {       // Adds a secondary bar object to the strategy.       AddDataSeries(BarsPeriodType.Minute, 5);   }   else if (State == State.DataLoaded)   {       // Syncs a Series object to the primary bar object       primarySeries = new Series<double>(this);               /\* Syncs another Series object to the secondary bar object.        We use an arbitrary indicator overloaded with an ISeries<double> input to achieve the sync.        The indicator can be any indicator. The Series<double> will be synced to whatever the        BarsArray[] is provided.\*/       secondarySeries = new Series<double>(SMA(BarsArray[1], 50));               // Stop-loss orders are placed 5 ticks below average entry price       SetStopLoss(CalculationMode.Ticks, 5);         // Profit target orders are placed 10 ticks above average entry price       SetProfitTarget(CalculationMode.Ticks, 10);   } } | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?multi-time_frame__instruments.htm#CreatingSeriesObjects)

tog_minus        [How Bars Data is Referenced](javascript:HMToggle('toggle','HowBarsDataIsReferenced','HowBarsDataIsReferenced_ICON'))

|  |
| --- |
| Understanding how multi-time frame bars are processed and which OHLCV data is referenced is critical.    Figure 1 below demonstrates the concept of bar processing on historical data or in real-time when the [Calculate](https://ninjatrader.com/support/helpGuides/nt8/calculate.htm) property is set to**Calculate.OnBarClose**. The 1 minute bars in yellow will only know the OHLCV of the 3 minute bar in yellow. The 1 minute bars in cyan will only know the OHLCV data of the 3 minute bar in cyan. Take a look at "Bar #5," which is the fifth one minute bar. If you wanted to know the current high value for the 3-minute time frame, you would get the value of the first 3 minute bar since this is the last "closed" bar. The second 3 minute bar (cyan) would not be known at that time.    Tips_3  Figure 1.  Bar processing on historical data using Calculate.OnBarClose  1.Primary 1-minute bar series  2.Secondary 3-minute bar series  3.Bar #5    Contrast the above image and concept with the image below, which demonstrates bar processing in real-time when the **Calculate** property is set to **Calculate.OnEachTick**  (tick by tick processing) or **Calculate.OnPriceChange (**processing by change in price). The 1 minute bars in yellow will know the current OHLCV of the 3 minute bar in yellow (second 3 minute bar) which is still in formation and has not yet closed.    Tips_4  Figure 2.  Bar processing in real-time using Calculate.OnEachTick or Calculate.OnPriceChange  1.Primary 1-minute bar series  2.Secondary 3-minute bar series  3.Bar #5    If you have a multi-time frame script in real-time, and it is processing tick by tick instead of on the close of each bar, understand that the OHLCV data you access in real-time is different than on historical data.    Below is another example to illustrate this point:    Your script has complex logic that changes the bar color on the chart. You are running tick by tick, as per the above "Figure 2" image, the 5th 1 minute bar is looking at OHLCV data from the second 3 minute bar. Your script changes the fifth 1 minute bar color to green. In the future, you reload your script into the chart and the fifth 1 minute bar is now a historical bar. As per Figure 1 above, the fifth 1 minute bar now references the OHLCV data of the first 3 minute bar (instead of the 2nd 3 minute bar as per Figure 2) and as a result, your script logic condition for coloring the bar green is no longer valid. The result is that now your chart looks different.    Special considerations for session boundaries :    Bars are not considered closed until the first tick of the following bar comes in (see also "True Event Driven OnBarUpdate" below). As a consequence, if the above series 2 cyan bar represents the final bar of a session, and this bar is referenced from the matching series 1 cyan bar, or anywhere after that, the data from the close of the bar (the beginning of the next session) will be referenced. If you plan on using multiple session templates, you will need to handle the final bar of a trading day case explicitly (for example, using a [Session Iterator](https://ninjatrader.com/support/helpGuides/nt8/sessioniterator.htm) and the [PriorDayOHLC](https://ninjatrader.com/support/helpGuides/nt8/prior_day_ohlc.htm)) if you would like to reference data from the end of the previous trading day instead of the beginning of the current trading day. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?multi-time_frame__instruments.htm#HowBarsDataIsReferenced)

tog_minus        [Using Bars Objects as Input to Indicator Methods](javascript:HMToggle('toggle','UsingBarsObjectsAsInputToIndicatorMethods','UsingBarsObjectsAsInputToIndicatorMethods_ICON'))

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| In the sub-section above, the concept of index values was introduced. This is a critical concept to understand since it is used consistently when working with multi-Bars script.    Let's demonstrate this concept:    Carrying on from the example above, our primary Bars is set from a MSFT 1 minute chart    **MSFT 1 minute Bars is given an index value of 0 in BarsArray**    In the OnStateChange() method we added a MSFT 3 minute Bars object and an AAPL 1 minute Bars object to the script    **MSFT 3 minute Bars is given an index value of 1 in BarsArray**  **AAPL 1 minute Bars is given an index value of 2 in BarsArray**    Incremental index values are given to Bars objects as they are added to a script. If there are 10 Bars objects in a script, then you will have index values ranging from 0 to 9.    Our script now has 3 Bars objects in the container **BarsArray**. From this point forward, we can ask this container to give us the Bars object we want to work with by providing the index value. The syntax for this is:    **BarsArray[index]**    This allows us to get the correct Bars object and use it as input for an [indicator method](https://ninjatrader.com/support/helpGuides/nt8/indicators.htm). For example:    **ADX(14)[0] > 30 && ADX(BarsArray[2], 14)[0] > 30**    The above expression in English would translate to:    **If the 14 period ADX of MSFT 1 minute is greater than 30 and the 14 period ADX of AAPL 1 minute is greater than 30**    Before we can apply this concept, we need to ensure that our Bars objects actually contain bars that we can use to run calculations. This can be done by checking the [CurrentBars](https://ninjatrader.com/support/helpGuides/nt8/currentbars.htm) array, which returns the number of the current bar in each Bars object. Using this in conjunction with [BarsRequiredToPlot](https://ninjatrader.com/support/helpGuides/nt8/barsrequiredtoplot.htm) will ensure each Bars object has sufficient data before we begin processing.     |  | | --- | | **Note**:  By default, the **CurrentBars** starting value will be -1 until all series have processed the first bar. |      | ns | | --- | | protected override void OnBarUpdate() {     // Checks to ensure all Bars objects contain enough bars before beginning.     // If this is a strategy, use BarsRequiredToTrade instead of BarsRequiredToPlot     if (CurrentBars[0] <= BarsRequiredToPlot || CurrentBars[1] <= BarsRequiredToPlot || CurrentBars[2] <= BarsRequiredToPlot)         return; } |     Putting it all together now, the following example checks if the current CCI value for all Bars objects is above 200. You will notice that BarsInProgress is used. This is to check which Bars object is calling the OnBarUpdate() method. More on this later in this section.     | ns | | --- | | protected override void OnBarUpdate() {     // Checks to ensure all Bars objects contain enough bars before beginning     // If this is a strategy, use BarsRequiredToTrade instead of BarsRequiredToPlot     if (CurrentBars[0] <= BarsRequiredToPlot || CurrentBars[1] <= BarsRequiredToPlot || CurrentBars[2] <= BarsRequiredToPlot)         return;       if (BarsInProgress == 0)     {         if (CCI(20)[0] > 200 && CCI(BarsArray[1], 20)[0] > 200           && CCI(BarsArray[2], 20)[0] > 200)         {               // Do something         }     } } | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?multi-time_frame__instruments.htm#UsingBarsObjectsAsInputToIndicatorMethods)

tog_minus        [True Event Driven OnBarUpdate() Method](javascript:HMToggle('toggle','TrueEventDrivenOnbarupdateMethod','TrueEventDrivenOnbarupdateMethod_ICON'))

|  |  |  |  |
| --- | --- | --- | --- |
| Since a NinjaScript script is truly event driven, the OnBarUpdate() method is called for every bar update event for each Bars object added to the script. This model maximizes flexibility. For example, you could have multiple trading systems combined into one strategy, each dependent on one another. For example, you could have a 1 minute MSFT Bars object and a 1 minute AAPL Bars object, process different trading rules on each Bars object and check to see if MSFT is long when AAPL trading logic is being processed.    The [BarsInProgress](https://ninjatrader.com/support/helpGuides/nt8/barsinprogress.htm) property is used to identify which Bars object is calling the OnBarUpdate() method. This allows you to filter out the events that you are looking for.    Continuing our example above, let's take a closer look at what is happening. Remember, we have three Bars objects working in our script, a primary Bars MSFT 1 minute, an MSFT 3 minute, and an AAPL 1 minute.     | ns | | --- | | protected override void OnBarUpdate() {     // Checks to ensure all Bars objects contain enough bars before beginning     // If this is a strategy, use BarsRequiredToTrade instead of BarsRequiredToPlot     if (CurrentBars[0] <= BarsRequiredToPlot || CurrentBars[1] <= BarsRequiredToPlot || CurrentBars[2] <= BarsRequiredToPlot)         return;       // Checks if OnBarUpdate() is called from an update on the primary Bars     if (BarsInProgress == 0)     {         if (Close[0] > Open[0])               // Do something     }       // Checks if OnBarUpdate() is called from an update on MSFT 3 minute Bars     if (BarsInProgress == 1)     {         if (Close[0] > Open[0])               // Do something     }       // Checks if OnBarUpdate() is called from an update on AAPL 1 minute Bars     if (BarsInProgress == 2)     {         if (Close[0] > Open[0])               // Do something     } } |     What is important to understand in the above sample code is that we have "if" branches that check to see what Bars object is calling the OnBarUpdate() method in order to process relevant trading logic. If we only wanted to process the events from the primary Bars we could add the following condition at the top of the OnBarUpdate() method:    **if (BarsInProgress != 0)** **return;**    What is also important to understand is the concept of context. When the OnBarUpdate() method is called, it will be called within the context of the calling Bars object. This means that if the primary Bars triggers the OnBarUpdate() method, all indicator methods and price data will point to that Bars object's data. Notice how the statement "if (Close[0] > Open[0]" exists under each "if" branch in the code sample above. The values returned by Close[0] and Open[0] will be the close and open price values for the calling Bars object. So when the BarsInProgress == 0 (primary Bars) the close value returned is the close price of the MSFT 1 minute bar. When the BarsInProgress == 1 the close value returned is the close price of the MSFT 3 minute Bars object.     |  | | --- | | **Notes:**  •A multi-series script only processes bar update events from the primary Bars (the series the script is applied to) and any additional Bars objects the script adds itself. Additional Bars objects from a multi-series chart or from other multi-series scripts that may be running concurrently will not be processed by this multi-series script.    •If a multi-series script adds an additional Bars object that already exists on the chart, the script will use the preexisting series instead of creating a new one to conserve memory. This includes that series' [session template](https://ninjatrader.com/support/helpGuides/nt8/using_the_trading_hours_window.htm) as applied from the chart. If the Bars object does not exist on the chart, the session template of the added Bars object will be the session template of the primary Bars object. If the primary Bars object is using the "<Use instrument settings>" session template, then the additional Bars objects will use the default session templates as defined for their particular instruments in the [Instruments](https://ninjatrader.com/support/helpGuides/nt8/instruments.htm) window.    •In a multi-series script, **CurrentBars** starting value will be -1 until all series have processed the first bar. To ensure you have satisfied this requirement on all your Bars objects, it is recommend you start your OnBarUpdate() method with [CurrentBars](https://ninjatrader.com/support/helpGuides/nt8/currentbars.htm) checks, as seen in the code sample above.    •A multi-series indicator will hold the same number of data points for plots as the primary series. Setting values to plots should be done in the primary series in OnBarUpdate(). If you are using calculations based off of a larger secondary series, it may plot like a step ladder because there are more data points available than there are actual meaningful data values.    •The default [CloseStrategy()](https://ninjatrader.com/support/helpGuides/nt8/closestrategy.htm) handling will only be applied to the primary series of a MultiSeries NinjaScript strategy.    •An indicator / strategy with multiple DataSeries of the same instrument will only process realtime OnBarUpdate() calls when a tick occurs in session of the trading hour templates of all added series.  Any ticks not processed will be queued and processed as a tick comes in for all subsequent DataSeries. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?multi-time_frame__instruments.htm#TrueEventDrivenOnbarupdateMethod)

tog_minus        [Accessing the Price Data in a Multi-Bars NinjaScript](javascript:HMToggle('toggle','AccessingThePriceDataInAMultibarsninjascript','AccessingThePriceDataInAMultibarsninjascript_ICON'))

|  |  |  |
| --- | --- | --- |
| As you probably know already, you can access the current bar's closing price with the following statement:    **Close[0];**    You can also access price data such as the close price of other Bars objects at any time. This is accomplished by accessing the [Opens](https://ninjatrader.com/support/helpGuides/nt8/opens.htm), [Highs](https://ninjatrader.com/support/helpGuides/nt8/highs.htm), [Lows](https://ninjatrader.com/support/helpGuides/nt8/lows.htm), [Closes](https://ninjatrader.com/support/helpGuides/nt8/closes.htm), [Volumes](https://ninjatrader.com/support/helpGuides/nt8/volumeseries.htm), [Medians](https://ninjatrader.com/support/helpGuides/nt8/medians.htm), [Typicals](https://ninjatrader.com/support/helpGuides/nt8/typicals.htm) and [Times](https://ninjatrader.com/support/helpGuides/nt8/timeseries.htm) series by index value. These properties hold collections (containers) that hold their named values for all Bars objects in a script.    Continuing with our example code above, if you wanted to access the high price of the MSFT 3 minute Bars object at index 1 you would write:   **Highs[1][0];**  This is just saying "give me the series of high prices for the Bars object at index 1 'Highs[1]' and return to me the current high value '[0]'". If the BarsInProgress index was equal to 1, the current  context is of the MSFT 3 min Bars object so you could just write:   **High[0];**    The following example demonstrates various ways to access price data.     | ns | | --- | | protected override void OnBarUpdate() {     // Checks to ensure all Bars objects contain enough bars before beginning     // If this is a strategy, use BarsRequiredToTrade instead of BarsRequiredToPlot     if (CurrentBars[0] <= BarsRequiredToPlot || CurrentBars[1] <= BarsRequiredToPlot || CurrentBars[2] <= BarsRequiredToPlot)         return;       // Checks if OnBarUpdate() is called from an update on the primary Bars     if (BarsInProgress == 0)     {         double primaryClose = Close[0];         double msft3minClose = Closes[1][0];         double aapl1minClose = Closes[2][0];           // primaryClose could also be expressed as         // primaryClose = Closes[0][0];     }       // Checks if OnBarUpdate() is called from an update on MSFT 3 minute Bars object     if (BarsInProgress == 1)     {         double primaryClose = Closes[0][0];         double msft3minClose = Close[0];         double aapl1minClose = Closes[2][0];     } } | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?multi-time_frame__instruments.htm#AccessingThePriceDataInAMultibarsninjascript)

tog_minus        [Entering, Exiting and Retrieving Position Information](javascript:HMToggle('toggle','EnteringExitingAndRetrievingPositionInformation','EnteringExitingAndRetrievingPositionInformation_ICON'))

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| This section is relevant for NinjaScript strategies only. Entry and Exit methods are executed within the BarsInProgress context. Let's demonstrate with an example:     | ns | | --- | | protected override void OnBarUpdate() {     // Checks to ensure all Bars objects contain enough bars before beginning     // If this is an indicator, use BarsRequiredToPlot instead of BarsRequiredToTrade     if (CurrentBars[0] <= BarsRequiredToPlot || CurrentBars[1] <= BarsRequiredToPlot || CurrentBars[2] <= BarsRequiredToPlot)         return;       // Checks if OnBarUpdate() is called from an update on the primary Bars     if (BarsInProgress == 0)     {         // Submits a buy market order for MSFT         EnterLong();     }       // Checks if OnBarUpdate() is called from an update on AAPL 1 minute Bars object     if (BarsInProgress == 2)     {         // Submits a buy market order for AAPL         EnterLong();           // Submits a buy market for MSFT when OnBarUpdate() is called for AAPL         EnterLong(0, 100, "BUY MSFT");     } } |   As you can see above, orders are submitted for MSFT when BarsInProgress is equal to 0 and for AAPL when BarsInProgress is equal to 2. The orders submitted are within the context of the Bars object calling the OnBarUpdate() method and the instrument associated to the calling Bars object. There is one exception, which is the order placed for MSFT within the context of the OnBarUpdate() call for AAPL. Each order method has a variation that allows you to specify the BarsInProgress index value which enables submission of orders for any instrument within the context of another instrument.     |  | | --- | | **Notes**:  1. Should you have multiple Bars objects of the same instrument while using Set() methods in your strategy, you should only submit orders for this instrument to the first Bars context of that instrument. This is to ensure your order logic is processed correctly and any necessary order amendments are done properly.    2. Should you have multiple Bars objects of the same instrument while using options to terminate orders/positions at the end of the session (TIF=Day or [IsExitOnSessionCloseStrategy](https://ninjatrader.com/support/helpGuides/nt8/isexitonsessionclosestrategy.htm)=true), you should not submit orders to Bars objects other than the first Bars context for that instrument when on the last bar of the session. This is necessary because some of the end of session handling is applied only to the first Bars context of an instrument, and submitting orders to other Bars objects for that instrument can bypass the end-of-session handling.    3.  For [advanced order methods](https://ninjatrader.com/support/helpGuides/nt8/advanced_order_handling.htm), if you **DO NOT** specify a BarsInProgress , the order will be submitted to the current bars in progress updating.  If the current BarsInProgress is a higher time frame, this could delay the time that the order is filled during historical backtesting.  As a result, you should always submit historical orders to the most granular of time frames.    4. When backtesting and submitting orders 'On bar close' and utilizing OnExecutionUpdate or OnOrderUpdate to submit orders, these orders will be processed immediately and filled by the fill engine depending on if the order satisfies its fill condition. This evaluation is done by looking ahead to the next bar of the current series. This is done prior to any secondary higher granularity series having a chance to run its 'OnBarUpdate' logic. If you planned on running order logic in your highest granularity added series then please insure that you submit orders in all cases to the highest granularity series. |     The [Position](https://ninjatrader.com/support/helpGuides/nt8/position.htm) property always references the position of the instrument of the current context. If the BarsInProgress is equal to 2 (AAPL 1 minute Bars), Position would refer to the position being held for AAPL. The [Positions](https://ninjatrader.com/support/helpGuides/nt8/positions.htm) property holds a collection of Position objects for each instrument in a strategy. Note that there is a critical difference here. Throughout this entire section we have been dealing with Bars objects. Although in our sample we have three Bars objects (MSFT 1 and 3 min and AAPL 1 min) we only have two instruments in the strategy.    **MSFT position is given an index value of 0** **AAPL position is given an index value of 1**    In the example below, when the OnBarUpdate() method is called for the primary Bars we also check if the position held for AAPL is NOT flat and then enter a long position in MSFT. The net result of this strategy is that a long position is entered for AAPL, and then once AAPL is long, we go long MSFT.     | ns | | --- | | protected override void OnBarUpdate() {     // Checks to ensure all Bars objects contain enough bars before beginning     // If this is an indicator, use BarsRequiredToPlot instead of BarsRequiredToTrade     if (CurrentBars[0] <= BarsRequiredToPlot || CurrentBars[1] <= BarsRequiredToPlot || CurrentBars[2] <= BarsRequiredToPlot)         return;       // Checks if OnBarUpdate() is called from an update on the primary Bars     if (BarsInProgress == 0 && Positions[1].MarketPosition != MarketPosition.Flat)     {         // Submits a buy market order for MSFT         EnterLong();     }       // Checks if OnBarUpdate() is called from an update on AAPL 1 minute Bars     if (BarsInProgress == 2)     {         // Submits a buy market order for AAPL         EnterLong();     } } | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?multi-time_frame__instruments.htm#EnteringExitingAndRetrievingPositionInformation)

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) >  **NinjaScript Best Practices** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/code_breaking_changes.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/distribution.htm) |

There are some best practices to be aware of when developing NinjaScript classes. The following tables present a non-exhaustive list of considerations to keep in mind when designing and implementing your code.

|  |
| --- |
| **Note**:   NinjaTrader is multi-threaded and event driven. Always assume that any of the methods you implement in NinjaScript could be called from another thread. |

tog_minus        [State management practices](javascript:HMToggle('toggle','StateResourceManagement','StateResourceManagement_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Managing Resources**  The [OnStateChange(](https://ninjatrader.com/support/helpGuides/nt8/onstatechange.htm)) method is called anytime there has been a change of [State](https://ninjatrader.com/support/helpGuides/nt8/state.htm) and can be used to help you setup, manage, and destroy several types of resources.  Where these values are setup is highly dependent on the kind of resource you are using.  The section below will cover how to manage various resources throughout different states.    **Setting Default UI Property Grid values**  Reserve **State.SetDefaults** for defaulting any public properties you wish to have exposed on the UI property grid.   You should also use this State for setting default desired NinjaScript property behavior which can be overridden from the property grid (e.g. [Calculate](https://ninjatrader.com/support/helpGuides/nt8/calculate.htm), [IsOverlay](https://ninjatrader.com/support/helpGuides/nt8/isoverlay.htm), etc.).  For Plots and Lines you wish to configure, [AddPlot()](https://ninjatrader.com/support/helpGuides/nt8/addplot.htm), [AddLine()](https://ninjatrader.com/support/helpGuides/nt8/addline.htm) should also have their default values set during this State     |  | | --- | | **Why:**Public values of the NinjaScript object in **SetDefaults** are pushed to the UI property grid for an opportunity to change settings of your object. |      | ns **Best practice** | | --- | | protected override void OnStateChange() {   // these are the values that show up as default on the UI   if (State == State.SetDefaults)   {     Calculate = Calculate.OnPriceChange;     IsOverlay = false;       Period = 50;       AddPlot(Brushes.Blue, "Plot Value");     AddLine(Brushes.Gray, 100, "Threshold");   } } |     For public properties you do **NOT** wish exposed to the UI property grid, set the [Browsable](https://ninjatrader.com/support/helpGuides/nt8/browsableattribute.htm) attribute to false:     | ns **Best practice** | | --- | | [Browsable(false)] // prevents from showing up on the UI property grid public int Communicator { get; set; } |     On indicators, properties you wish to set from other objects, set the [NinjaScriptPropertyAttribute](https://ninjatrader.com/support/helpGuides/nt8/ninjascriptpropertyattribute.htm):     | ns **Best practice** | | --- | | [NinjaScriptProperty] // can now call MyIndicator(20) from another object public int Period { get; set; } |     The default behavior is to serialize any public properties and fields to a Workspace or Template file when saving. However, not all objects can be serialized - or you may wish to exclude a property from being saved and restored.  For these scenarios, set the [XMLIgnore](https://ninjatrader.com/support/helpGuides/nt8/xmlignoreattribute.htm) attribute to the property:     | ns **Best practice** | | --- | | [XmlIgnore] // removes from serialization     public Brush DownBrush { get; set; } |     As a best practice as well, your NinjaScript should not have any public fields, since those would get serialized as well - which means their state would be persisted, which in turn could lead to unexpected outcomes.     |  | | --- | | **Tip:**See the [Working with Brushes](https://ninjatrader.com/support/helpGuides/nt8/working_with_brushes.htm) section of the Help Guide for information on properly serializing brushes |     **Calculating run-time object values** Do not attempt to do advanced calculations or try to access object references in **State.SetDefaults**.  This State should be kept as lean as possible, and any calculation logic should be delayed until at least **State.Configure**     |  | | --- | | **Why:**Your object will be called in situations you may not be expecting. You can read more about this subject on [Understanding the life cycle of your NinjaScript objects](https://ninjatrader.com/support/helpGuides/nt8/understanding_the_lifecycle_of.htm) |      | ns **Practice to avoid** | | --- | | protected override void OnStateChange() {   if (State == State.SetDefaults)   {       // logic could take longer than desired as the list of indicator names is populated     for (int i = 0; i <= array.length; i ++)         DoWork(i);      // possible null reference exception since TickSize is not set yet     Period = 5 \* TickSize;   } } |      | ns **Best practice** | | --- | | protected override void OnStateChange() {   // Complex operations should be delayed to >= State.Configure   if (State == State.Configure)   {     for (int i = 0; i < = array.length; i ++)         DoWork(i);   }     // information related to market data is not available until at least State.DataLoaded   else if (State == State.DataLoaded)   {     Period = 5 \* TickSize;   } } |     **Setting class level variables**  Do not set variables at the class level unless they are constant.  You should delay setting or resetting variables until the **State** has reached **State.Configure**.  You can use const keyword to differentiate values which do not change from variables which do change.     |  | | --- | | **Why**:  Waiting to set up and define resources until the object has been configured ensures that values not set up and declared prematurely. |      | ns **Best practice** | | --- | | // value is always 5, it can be made constant and declared at the class level private const int multiplier = 5;   // these values can change, may be better to delay setting until State.Configure private int counter; private List<int> myList;  protected override void OnStateChange() {   if (State == State.Configure)   {     counter = 0;     myList = new List<int>();   }   } |     **Resetting class level variables for Strategy Analyzer Optimization**    To take advantage of performance optimizations, developers may need to reset class level variables in the strategy otherwise unexpected results can occur.     |  | | --- | | **Why**:  When optimizing a strategy, instances may or may not be recycled depending on the strategy [IsInstantiatedOnEachOptimizationIteration](https://ninjatrader.com/support/helpGuides/nt8/isinstantiatedoneachoptimizationiteration.htm) setting. |      | ns **Best practice** | | --- | | // examples of fields which need to be reset private double myDouble; private bool myBool; private DateTime myDateTime; private Order myOrderObject; private Brush myBrushObject; private Array myIntArray; private List<object> myList; private SMA mySMAIndicator; private Series<double> mySeries;   protected override void OnStateChange() {   if (State == State.SetDefaults)   {     // disabled to take advantage of performance gains     // However any strategy state that would be mutable after State.SetDefaults needed to be reset for the next run.     IsInstantiatedOnEachOptimizationIteration = false;   }   else if (State == State.Configure)   {     // Since these values are not dependent on bars, they can be reset as early as State.Configure     myDouble = double.MinValue;     myBool = false;     myDateTime = DateTime.MinValue;     myOrderObject = null;     myBrushObject = null;       if (myIntArray != null)         Array.Clear(myIntArray, 0, myIntArray.Length);     else         myIntArray = new int[20];       if (myList != null)         myList.Clear();     else         myList = new List<object>();   }     else if (State == State.DataLoaded)   {     // Since these values do are dependent on bars, they should only reset during State.DataLoaded     mySMAIndicator = SMA(14);     mySeries = new Series<double>(this);   } } |     **Accessing properties related to market data**  Do not attempt to access objects related to instrument market data until the **State** has reached **State.DataLoaded**     |  | | --- | | **Why**: Waiting to access objects that depend on market data until **DataLoaded** prevents access errors in all scenarios |      | ns **Best practice** | | --- | | protected override void OnStateChange() {   if (State == State.DataLoaded)   {     // these objects and their related members are not available until State.DataLoaded     Print(Bars.Count);     Print(Instrument.FullName);     Print(BarsPeriod.BarsPeriodType);     Print(TradingHours.TimeZone);     Print(Input);   } } |      |  | | --- | | **Note**: All additional data series must be added in **State.Configure**(this includes series that any hosted script potentially needs as well - [more info](http://ninjatrader.com/support/helpGuides/nt8/en-us/adddataseries.htm)). Since objects such as [Instrument](https://ninjatrader.com/support/helpGuides/nt8/instrument.htm), [BarsPeriod](https://ninjatrader.com/support/helpGuides/nt8/barsperiod.htm), [TradingHours](https://ninjatrader.com/support/helpGuides/nt8/tradinghours.htm), etc. are **NOT** guaranteed to be available until **State.DataLoaded**, you cannot reliably use the primary instrument properties as arguments in [AddDataSeries()](https://ninjatrader.com/support/helpGuides/nt8/adddataseries.htm).  Attempting to add a data series dynamically is **NOT** guaranteed and therefore should be avoided.  In some cases, you may be able to use a [BarsRequest()](https://ninjatrader.com/support/helpGuides/nt8/barsrequest.htm) to obtain market data for other instruments and intervals. |     **Setting up resources that rely on market data**  For objects which depend on market data, delay their construction until the **State** has reached **State.DataLoaded**     |  | | --- | | **Why**: Waiting to construct objects that depend on market data until **DataLoaded** ensures that their underlying input contains significant values in all scenarios. |      | ns **Best practice** | | --- | | // these resources depend on bars, wait until State.DataLoaded to instantiated private EMA myEMA; private Series<double> mySeries; private SessionIterator mySessionIterator;   protected override void OnStateChange() {     if (State == State.DataLoaded)   {     myEMA = EMA(20);     mySeries = new Series<double>(this);     mySessionIterator = new SessionIterator(Bars);   } } |     **Accessing element on the UI**  For objects which exist on the UI (e.g., [ChartControl](https://ninjatrader.com/support/helpGuides/nt8/chartcontrol.htm), [ChartPanel](https://ninjatrader.com/support/helpGuides/nt8/chartpanel.htm), [ChartBars](https://ninjatrader.com/support/helpGuides/nt8/chartbars.htm), [NTWindow](https://ninjatrader.com/support/helpGuides/nt8/ntwindow.htm), etc.) wait until the State has reached State.Historical.  This practice is correct for both reading properties or should you wish to add custom elements to the existing UI.     |  | | --- | | **Why**:  NinjaTrader UI related objects are not guaranteed to be available until historical data processing has started. |      | ns **Best practice** | | --- | | protected override void OnStateChange() {   // wait until at least State.Historical   if (State == State.Historical)   {     // and double check UI object is not null before accessing     if (ChartControl != null)     {         Print(ChartControl.Properties.ChartBackground);     }   } } |     **Transitioning order references from historical to real-time**  When dealing with strategy based orders which have transitioned from historical to real-time, you will need to ensure that locally stored order references are also updated.     |  | | --- | | **Why**: As the core order object updates, NinjaTrader has no specific way to update your locally stored order references.  You can read more about this subject on the Advanced Order Handling topic: [Transitioning order references from historical to live](https://ninjatrader.com/support/helpGuides/nt8/advanced_order_handling.htm) |      | ns **Best practice** | | --- | | private Order entryOrder = null;   protected override void OnBarUpdate() {   if (entryOrder == null && Close[0] > Open[0])     entryOrder = EnterLongLimit("myEntryOrder", Low[0]); }  protected override void OnOrderUpdate(Order order, double limitPrice, double stopPrice, int quantity, int filled, double averageFillPrice, OrderState orderState, DateTime time, ErrorCode error, string nativeError)  {    // One time only, as we transition from historical    // Convert any old historical order object references to the live order submitted to the real-time account    if (entryOrder != null && entryOrder.IsBacktestOrder && State == State.Realtime)        entryOrder = GetRealtimeOrder(entryOrder);       // Null entryOrder if filled or cancelled. We do not use the Order objects after the order is filled, so we can null it here    if (entryOrder != null && entryOrder == order)     {        if (order.OrderState == OrderState.Cancelled && order.Filled == 0)            entryOrder = null;        if (order.OrderState == OrderState.Filled)            entryOrder = null;     }  } |     **Terminating custom resources**  Use a flag to track when resources have been set up properly before attempting to destroy them.     |  | | --- | | **Why**:  Checking that an object has been configured ensures that values not destroyed prematurely. You can read more about this subject on [Understanding the life cycle of your NinjaScript objects](https://ninjatrader.com/support/helpGuides/nt8/understanding_the_lifecycle_of.htm) |      | ns **Best practice** | | --- | | protected override void OnStateChange() {   if (State == State.Configure)   {     myObject = new object();     // set a flag to indicator object has been configured     configured = true;   }     else if (State == State.Terminated)   {     // only dispose of object if it has been configured     if (configured)     {         myObject.Dispose();     }   } } | |

tog_minus        [Error handling practices](javascript:HMToggle('toggle','Errorhandling','Errorhandling_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Safely accessing reference objects**  Although there are documented **States** where objects are available, the implementation could change.  If you are accessing a reference object, please do so by first checking that the object is not null.     | ns **Best practice** | | --- | | // checking to ensure chart control is available in all situations // will help to ensure this logic below does not generate errors at a later time if(ChartControl != null) {   myBackgroundBrush = ChartControl.Properties.ChartBackground; } |     **Accessing objects which terminate**  To protect against race conditions and access errors, you should temporarily check for reference errors any time you attempt to do something with an object.     |  | | --- | | **Why**: **OnStateChange()** runs asynchronous to other NinjaScript events.  You can run into scenarios where you **State.Terminated** logic is called in the middle of OnBarUpdate(), OnRender() etc. |      | ns **Best practice** | | --- | | protected override void OnStateChange() {   // this logic runs asynchronously to other events   if (State == State.Terminated)   {     myObject = null;   } } protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   if (myObject == null)     return;     // for safety, always check for null references before attempting to access an object   // even if you have once checked for null references earlier run-time   if (myObject != null)     myObject.DoSomething(); } |     **Proving instructions for non-ninjascript properties**  Do not attempt to modify existing UI "Properties" to meet your specific needs.  These features are exposed to allow you to read the environment state and make decisions to alter how your code executes, but should not be relied on to modify settings on behalf of the user.  While these objects from these classes have setters for technical reasons, you should not attempt to amend the values through code.  Instead, you should issue warnings or log errors instructing users to modify settings when required:     |  | | --- | | **Why**:  NinjaTrader makes no guarantee that the requested changes will take effect, and user settings always take precedences.  This includes the user defined [ChartControl.Properties](https://ninjatrader.com/support/helpGuides/nt8/chartcontrol_properties.htm), [ChartBars.Properties](https://ninjatrader.com/support/helpGuides/nt8/chartbars_properties.htm), and [ChartPanel.Properties](https://ninjatrader.com/support/helpGuides/nt8/chartscale_properties.htm). Furthermore, two different user scripts could be installed which also attempt to modify properties you are relying which could introduce conflicts. |      | ns **Best practice** | | --- | | if (State == State.Historical) {   if (ChartControl.Properties.EquidistantBarSpacing == true)   {     Draw.TextFixed(this, "error", "This indicator works best with Equidistant BarSpacing set to false.", TextPosition.BottomRight);   } } |     **Modifying UI elements and multi-threading**  When interacting with UI objects, such as obtaining UI information, or modifying the existing layout, always use the NinjaScript's Dispatcher asynchronously     |  | | --- | | **Critical**:  Improper thread handling from a NinjaScript object is a common cause of application deadlocks.  Please be sure to read more information on [Multi-Threading Consideration for NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/multi-threading.htm) |      | ns **Best practice** | | --- | | // using a Dispatcher will ensure that the corresponding action executes on the associated thread this.Dispatcher.InvokeAsync(() => {   UserControlCollection.Add(new System.Windows.Controls.TextBlock   {     Text = "\nAdded by the ChartControl Dispatcher."   }); }); |       **Properly implementing try/catch blocks**  Unless you are specifically debugging a method, the use of a try-catch block should be scoped to a particular area of logic.  Do **NOT** try to handle all of your execution logic under one giant try-catch block.     |  | | --- | | **Why**:  Larger try-catch blocks can not only be harder to debug, but can introduce performance issues at run-time |      | ns **Practice to avoid** | | --- | | protected override void OnBarUpdate() {   try   {     // encapsulates entire OnBarUpdate logic   }   catch (Exception ex)   {     // attempt to handle all errors in one catch   } } |     **Using WPF brushes**  Try to use a static predefined Brush if possible.  If you need to customize a new brush object, make sure to .Freeze() the brush before using it.     |  | | --- | | **Why**:  The pre-defined brushes are thread safe and do not require any special handling.  Custom defined brushes, on the other hand, are **NOT** thread-safe and must be frozen otherwise cross-thread exceptions can occur. |      | ns **Best practice** | | --- | | // predefined brush BackBrush = Brushes.Blue;   // if you are using a custom brush to e.g., modify the opacity SolidColorBrush opaqueBlue = new SolidColorBrush(Colors.Blue) {Opacity = .25f};   // or just using at custom color not available in pre-defined brushes class SolidColorBrush coolGreen = new SolidColorBrush(Color.FromRgb(30, 255, 128));   // you must freeze these brushes after they are constructed! opaqueBlue.Freeze(); coolGreen.Freeze(); |     **barsAgo indexer vs. absolute bar Index**  As you probably know, you can quickly look up the bar value on the chart by calling a [PriceSeries<T>](https://ninjatrader.com/support/helpGuides/nt8/priceseries.htm) barsAgo indexer, e.g., Close[0].  However, the internal indexer and pointers about the barsAgo value are only guaranteed to be correctly synced and updated during a market data event.  As a result, you should favor using the absolute [GetValueAt()](https://ninjatrader.com/support/helpGuides/nt8/getvalueat.htm) methods during events which are not driven by price     |  | | --- | | **Why**:  Attempting to call the barsAgo indexer in an event method that is not driven by market data can yield unexpected results. |      | ns **Best practice** | | --- | | // OnRender is not a market data event; barsAgo pointers are not guaranteed to be in sync protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   Print(mySMA.GetValueAt(CurrentBar)); }   // same is true for you custom events private void myCustomClickHandler(object sender, MouseButtonEventArgs e) {   Print(Close.GetValueAt(CurrentBar)); } |      |  | | --- | | **Tip**:  If you have programming requirements which rely on a PriceSeries indexer, you can use the [TriggerCustomEvent()](https://ninjatrader.com/support/helpGuides/nt8/triggercustomevent.htm) delegate which will update the internal pointers and indexes before executing the logic you specify. |     **Casting safely**  Avoid type casting and type conversion as much as possible.  Casting from a mixed collection of types is also prone to exceptions especially in situations that may not occur when you originally test your code.     |  | | --- | | **Why**:   The practice to avoid code below could work in some scenarios but would generate errors if other types were added to that collection that you were not anticipating. |      | ns **Practice to avoid** | | --- | | // This would run without errors if there were \_ONLY\_ type HoriztonalLine on the chart // But you risk a likely 'System.InvalidCastException' when other draw types are in that collection foreach (HorizontalLine hLine in DrawObjects) {   } |     If you must cast, do so safely and avoid implicit casts to types which may not be guaranteed to succeeded     | ns **Best practice** | | --- | | // Use the base IDrawingTool type and then cast to the desired type within the for loop foreach (IDrawingTool hLine in DrawObjects) {   // Note:  to prevent further errors, your type casting should be done using the "as" keyword   // Opposed to a direct cast:   // HorizontalLine myLine = (HorizontalLine) hLine;    HorizontalLine myLine = hLine as HorizontalLine;     // This will allow you to ensure the cast actually occurred   if (myLine != null)   {     Print(myLine.StartAnchor.Price);   } } | |

tog_minus        [Performance practices](javascript:HMToggle('toggle','Performance','Performance_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Referencing indicator methods**  In general, when calling an Indicator return method, there is some internal caching which occurs by design to help reduce memory consumption.     |  | | --- | | **Why**:  While the designed indicator caching improves general memory performance, there is an implied cost of actually looking up the cached indicator |      | ns **Practice to avoid** | | --- | | // each time you call the SMA() return method there is a small performance cost // implied from the time it takes to look up the cached instance if (Close[0] > SMA(20)[0]) {   Print(SMA(20)[0]);   EnterLongLimit(SMA(20)[0]);   Draw.Dot(this, Time[0].ToString(), false, 0, SMA(20)[0], Brushes.DarkGreen); } |      |  | | --- | | **Note**:  Indicator caching **ONLY**occurs when an indicator is recalled with the same **EXACT** parameters and input from the **SAME** calling script. (i.e. when a previously called indicator is called a second time with new parameters in the same script, a second instance will be created / cached) |     If you are reusing an indicator several times through your code (especially indicators with many parameters), you can take further steps to refine performance by storing a reference to the indicator instance yourself (although it is by no means a requirement, and this suggestion does not need to be followed strictly)     | ns **Best practice** | | --- | | private SMA mySma;   protected override void OnStateChange() {   // when the indicator begins processing   // save an instance of the SMA indicator with the desired input     if (State == State.Historical)   {     mySma = SMA(20);   } }   protected override void OnBarUpdate() {   // use the referenced mySMA throughout the lifetime of the script   if (Close[0] > mySma[0])   {     Print(mySma[0]);     EnterLongLimit(mySma[0]);     Draw.Dot(this, Time[0].ToString(), false, 0, mySma[0], Brushes.DarkGreen);   } } |     **Marking object references for garbage collection**  While it is not always necessary to set objects to null, doing so will mark them for garbage collection sooner and help prevent unnecessary memory resources from being utilized.     |  | | --- | | **Why**:   In general you should be diligent to set stored memory objects to null when you are done using them, especially in situations where a NinjaScript object may be running for an extended period. |      | ns **Best practice** | | --- | | protected override void OnBarUpdate() {   // saving "myDot" creates an additional reference in memory   Dot myDot = Draw.Dot(this, "myDot" + CurrentBar, false, Time[0], Close[0], Brushes.Blue);     if (conditionToRemove)   {     // remove draw object will remove the object from the chart     RemoveDrawObject("myDot");       // but your local object "myDot" is still stored in memory.     // Explicitly setting to null will ensure object is marked for garbage collection     myDot = null;   } } |      |  | | --- | | **Note**:  The example above demonstrates using a draw object, but the practice can be extended to any object you store in memory (e.g., orders, brushes, custom objects, etc) |     **Disposing of custom resources**  Dispose of objects that inherit from IDisposable or put into a Using statement.     |  | | --- | | **Why**:  NinjaTrader is not guaranteed to dispose of objects for you.  To avoid unnecessary memory consumption, always manage your resources by creating a variable and dispose of the object. |      | ns **Best practice** | | --- | | // example of object instantiated which need to be disposed StreamWriter writer = new StreamWriter("some\_file.txt");   // use the object writer.WriteLine("Some text");   // implements IDisposbile, make sure to call .Dispose() when finished writer.Dispose();   // or put in "using" statement which implicitly calls .Dispose() when finished using (StreamWriter writer2 = new StreamWriter("some\_file.txt")) {   writer2.WriteLine("Some text"); } |      |  | | --- | | **Tip**:  This is most commonly applicable when using SharpDX resources for custom rendering.  Please be sure to review the information on [Best Practices for SharpDX Resources](https://ninjatrader.com/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm#bestpracticesforsharpdxresources) |     **Avoiding duplicate calculations**  Be mindful where and when your potentially complex calculations would be recalculated and thus run the risk of being calculated redundantly. For example, you may have logic which only needs to calculate, e.g., once per instance, once per session, once per bar, etc.     | ns **Best practice** | | --- | | // get GetPreviousTradingDayEnd() is expensive to look up // but value only needs to be looked up once a day -> only calcualte on first bar of session if (Bars.IsFirstBarOfSession) {   TradingHours.GetPreviousTradingDayEnd(Time[0]); } |     The same considerations would apply to variables or function calls that would not change their output value for the currently processed bar on [Calculate.OnEachTick](https://ninjatrader.com/support/helpGuides/nt8/calculate.htm) or [.OnPriceChange](https://ninjatrader.com/support/helpGuides/nt8/calculate.htm), thus there would be no need handling them outside of [IsFirstTickOfBar](https://ninjatrader.com/support/helpGuides/nt8/isfirsttickofbar.htm)     | ns **Best practice** | | --- | | // dedicated logic to cache the prior sum on each tick of bar // While it is a good practice, this can cause problems for bar types which may remove last bar (see below) if (IsFirstTickOfBar)   priorSum = sum;   sum = priorSum + Input[0] - (CurrentBar >= Period ? Input[Period] : 0); Value[0] = sum / (CurrentBar < Period ? CurrentBar + 1 : Period); |     **Caching values on bars which remove last bar**  Building on the previous example, be careful when caching values on the first tick of bar if using bars types which are [IsRemoveLastBarSupported](https://ninjatrader.com/support/helpGuides/nt8/isremovelastbarsupported.htm).  To see how to handle these situations best, take a look at the default SMA indicator which has an additional logic branch which disables caching on those bar types:     | ns **Best practice** | | --- | | // logic below disables first tick of bar caching only on bar types which remove last bar if (BarsArray[0].BarsType.IsRemoveLastBarSupported) {   if (CurrentBar == 0)     Value[0] = Input[0];   else   {     double last = Value[1] \* Math.Min(CurrentBar, Period);       if (CurrentBar >= Period)         Value[0] = (last + Input[0] - Input[Period]) / Math.Min(CurrentBar, Period);     else         Value[0] = ((last + Input[0]) / (Math.Min(CurrentBar, Period) + 1));   } } |     **Precomputing values instead of calculating in OnRender()**  To preserve good performance, always err on the side of caution if you are using OnRender for any calculation logic.     |  | | --- | | **Why**:   OnRender() is called frequently as you interact with the Chart, which can cause calculations to occur much more often than the related market data events and can cause unnecessary spikes in CPU consumption. |      | ns **Practice to avoid** | | --- | | protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {  // continually recalling the same value methods is unnecessary in this situation       double myValue = Bars.GetClose(CurrentBar) + Bars.GetOpen(CurrentBar);     // render myValue } |      | ns **Best practice** | | --- | | private double myValue;  protected override void OnBarUpdate() {   // myValue only needs to update when OnBarUpdate() is called   // and then can be passed to OnRender() for chart rendering purposes   myValue = Close[0] + Open[0]; }   protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // if needed, you can always check that myValue has actually been set   if (myValue > double.MinValue)   {     // render myValue   } } |     **Restricting OnRender() calculations to visible ChartBars**  Use the [ChartBars.FromIndex](https://ninjatrader.com/support/helpGuides/nt8/chartbars_fromindex.htm) and [ChartBars.ToIndex](https://ninjatrader.com/support/helpGuides/nt8/chartbars_toindex.htm) to limit calculations to only what is visible on the chart     |  | | --- | | **Why:**Rendering should be reserved for rendering on what is visible on the Chart.  Performing calculations on bar index which are not visible can cause random spikes in CPU consumption. |      | ns **Best practice** | | --- | | protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // restricting this loop to only the ChartBars.From/ToIndex limits the loop to only what is visible on the chart   for (int barIndex = ChartBars.FromIndex; barIndex <= ChartBars.ToIndex; barIndex++)   {     Print(ChartControl.GetSlotIndexByX(barIndex));   } } |     **Using DrawObjects vs custom graphics in OnRender()**  When using [Draw methods](https://ninjatrader.com/support/helpGuides/nt8/drawing.htm), a new instance of the Draw object is created including its custom rendering and calculation logic.  These methods are convenient in many situations, but can quickly introduce performance issues if used too liberally.  In some situations, you may see better performance for rendering via [SharpDX](https://ninjatrader.com/support/helpGuides/nt8/sharpdx.htm) in [OnRender()](https://ninjatrader.com/support/helpGuides/nt8/onrender.htm).     |  | | --- | | **Why**: Each draw object instance will see its own OnRender() called to render values. If you instead implement custom rendering in the your object, you would only see a single OnRender() call for your custom created graphics. |      | ns **Practice to avoid** | | --- | | protected override void OnBarUpdate() {   // this would draw a dot on every bar on the chart   // each instance would need to call its own OnRender() method   // not a very efficient use a draw method   Draw.Dot(this, "everyDot" + CurrentBar, false, 0, Close[0], Brushes.Blue); } |     With just a little extra code (much less than what is in the Draw methods) custom SharpDX rendering greatly reduces CPU and Memory consumption    **Please ensure** a Direct2D1 factory would only be instantiated from [OnRender()](https://ninjatrader.com/support/helpGuides/nt8/onrender.htm) or [OnRenderTargetChanged()](https://ninjatrader.com/support/helpGuides/nt8/onrendertargetchanged.htm) (which run in the UI thread), as access from other threads outside those methods could cause a degradation in performance.     | ns **Best practice** | | --- | | protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // achieves the same effect of drawing a dot on every bar   // but only needs to call your object's OnRender()   for (int index = ChartBars.FromIndex; index <= ChartBars.ToIndex; index++)   {     float price = chartScale.GetYByValue(Close.GetValueAt(index));     float bar = chartControl.GetXByBarIndex(ChartBars, index);     float radius = (float) chartControl.BarWidth;       SharpDX.Direct2D1.Ellipse dot = new SharpDX.Direct2D1.Ellipse(new SharpDX.Vector2(bar, price), radius, radius);       using (SharpDX.Direct2D1.SolidColorBrush brush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.Blue))     {         RenderTarget.FillEllipse(dot, brush);     }   } } |      |  | | --- | | **Tip**:  One of the advantages of using a Draw.Method is the returned Draw Objects contains metadata which could be used later (such as for obtain the bar index or price value of the dot later on).  If you would use this metadata later on, using a Draw method would be in your best interests.  However, if you are solely looking to render figures on a chart, favoring your custom SharpDX methods can drastically improve performance. |     **Responding to user events**  Do **NOT** use OnRender() for purposes other than rendering.  If you need events to hook into user interactions, consider adding your own event handler.  The example below shows registering the ChartPanel MouseDown event and registering a custom WPF control     |  | | --- | | **Why:**OnRender() may call more or less frequently than you anticipated.  Using your own custom event handlers allows you control and isolate user event logic you are looking to capture | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/support/helpGuides/nt8/common.htm) >  **ISeries<T>** | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/url.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/common.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/seriest.htm) |

**Definition**

ISeries<T> is an interface that is implemented by all NinjaScript classes that manage historical data as an ISeries<double> (Open, High, Low, Close, etc), used for indicator input, and other object data.  Please see the help guide article on [Working with Price Series](https://ninjatrader.com/support/helpGuides/nt8/working_with_price_series.htm) for a basic overview on how to access this information.

**Types of ISeries**

|  |  |
| --- | --- |
| [Series<T>](https://ninjatrader.com/support/helpGuides/nt8/seriest.htm) | Represents a generic custom data structure for custom development |
| [PriceSeries](https://ninjatrader.com/support/helpGuides/nt8/priceseries.htm) | Historical price data structured as an ISeries<double> interface (Close[0], High[0], Low[0], etc) |
| [TimeSeries](https://ninjatrader.com/support/helpGuides/nt8/timeseries.htm) | Historical time stamps structured as an ISeries<DateTime> interface (Time[0]) |
| [VolumeSeries](https://ninjatrader.com/support/helpGuides/nt8/volumeseries.htm) | Historical volume data structured as an ISeries<double> interface (Volume[0]) |

**Methods and Properties**

|  |  |
| --- | --- |
| [GetValueAt()](https://ninjatrader.com/support/helpGuides/nt8/getvalueat.htm) | Returns the underlying input value at a specified bar index value. |
| [IsValidDataPoint()](https://ninjatrader.com/support/helpGuides/nt8/isvaliddatapoint.htm) | Indicates if the specified input is set at a barsAgo value relative to the current bar. |
| [IsValidDataPointAt()](https://ninjatrader.com/support/helpGuides/nt8/isvaliddatapointat.htm) | Indicates if the specified input is set at a specified bar index value. |
| [Count](https://ninjatrader.com/support/helpGuides/nt8/iseries_count.htm) | Return the number total number of values in the ISeries array |

|  |
| --- |
| **Tips**: (see examples below)  1.By specifying a parameter of type ISeries<double>, you can then pass in an array of closing prices, an indicator, or a user defined data series.  2.When working with ISeries<double> objects in your code you may come across situations where you are not sure if the value being accessed is a valid value or just a "placeholder" value. To check if you are using valid values for your logic calculations that have been explicitly set, please use .IsValidDataPoint(int *barsAgo*)to check. |

**Examples**

| ns **Using ISeries as a method parameter** |
| --- |
| //create custom a method named DoubleTheValue that accepts any object that implements // the ISeries<double> interface as a parameter private double DoubleTheValue(ISeries<double> priceData) {     return priceData[0] \* 2; }   protected override void OnBarUpdate() {   // This custom method is then used twice,   //the first time passing in an array of closing prices     Print(DoubleTheValue(Close));   //and the second time passing in a 20 period simple moving average.     Print(DoubleTheValue(SMA(20))); } |

| ns **Checking ISeries value before accessing** | |
| --- | --- |
| protected override void OnBarUpdate() {     // Only set our plot if the input is a valid value     if (Input.IsValidDataPoint(0))         Plot0[0] = Input[0]; } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/support/helpGuides/nt8/common.htm) > [ISeries<T>](https://ninjatrader.com/support/helpGuides/nt8/iseriest.htm) > [PriceSeries<double>](https://ninjatrader.com/support/helpGuides/nt8/priceseries.htm) >  **Input** | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/highs.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/priceseries.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/inputs.htm) |

**Definition**

The main historical data input. If implemented in the NinjaScript object, it allows for more flexibility as non bars based series such as plot series could be passed in and drive the calculation outcomes - an example would be a custom moving average that should have the ability to operate on another moving average (i.e. the SMA) as input series.

**Property Value**

An ISeries<double> type object that implements the Series<double> interface. Accessing this property via an index value [int *barsAgo*] returns a double value representing the price of the referenced bar.

**Syntax**

Input  
Input[int *barsAgo*]

**Examples**

| ns |
| --- |
| // Prints the the current value of input Print(Input[0].ToString()); |

| ns |
| --- |
| // Prints the the current type of input passed to the object, so we can detect if we're working on a price based series such as OHLCV or a derivative such as an SMA indicator if (Input is PriceSeries) Print("Price Series Input"); if (Input is Indicator) Print("Indicator Input"); |

| ns |
| --- |
| // Prints the the current selected price type for the input series else if (State == State.DataLoaded) {     PriceSeries priceSeries = Inputs[0] as PriceSeries;                 if (priceSeries != null)         Print("PriceType selected: " + priceSeries.PriceType); } |

|  |  |
| --- | --- |
| **Tip**: When working with multi-series indicators, Input is not guaranteed to reference the primary [BarsInProgress](https://ninjatrader.com/support/helpGuides/nt8/barsinprogress.htm). Please be mindful as to when you access Input[0] as you will only be able to do so after the contextual BarsInProgress has bars. To check to ensure BarsInProgress has some bars you can use [CurrentBars](https://ninjatrader.com/support/helpGuides/nt8/currentbars.htm) to check. | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) > [Educational Resources](https://ninjatrader.com/support/helpGuides/nt8/educational_resources.htm) >  **Working with Price Series** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/working_with_pixel_coordinates.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/educational_resources.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/reference_samples.htm) |

**Price Data Overview**

The core objective of developing custom Indicators and Strategies with NinjaScript is to evaluate price data. NinjaScript allows you to reference current and historical price data. There are several categories of price data which include ISeries<T>, Indicator and Custom Historical Series.

**Definitions**

|  |  |
| --- | --- |
| [ISeries<T>](https://ninjatrader.com/support/helpGuides/nt8/priceseries.htm) | Standard bar based price types such as closing, opening, high, low prices and volume |
| [Indicator](https://ninjatrader.com/support/helpGuides/nt8/indicator.htm) | Calculated values based on price type values such as a simple moving average |
| Custom Historical [Series<T>](https://ninjatrader.com/support/helpGuides/nt8/seriest.htm) | Custom calculated values that you wish to store and associate to each historical bar |

**Referencing Series**

|  |  |  |  |
| --- | --- | --- | --- |
| **ISeries<T>** | **Syntax** | **Editor Shortcut** | **Definition** |
| Close | Close[int *barsAgo*] | "c" + Tab Key | Last traded price of a bar |
| Open | Open[int *barsAgo*] | "o" + Tab Key | Opening price of a bar |
| High | High[int *barsAgo*] | "h" + Tab Key | Highest traded price of a bar |
| Low | Low[int *barsAgo*] | "l" + Tab Key | Lowest traded price of a bar |
| Volume | Volume[int *barsAgo*] | "v" + Tab Key | Number of shares/contracts traded of a bar |
| Input | Input[int *barsAgo*] | "i" + Tab Key | Default price type of a bar |

You will notice that to reference any price data you need to include a value for [int *barsAgo*]. This is a very simple concept; barsAgo represents the number of bars ago to reference and int indicates that barsAgo is an integer value. As an example, we could write a statement to check if the the high price of 1 bar ago is less than the high price of the current bar like this:

 High[1] < High[0];

You could write a statement to calculate the average closing price of the last three bars like this:

 ( Close[2] + Close[1] + Close[0] ) / 3;

As you may have already figured out, referencing the current bar data is accomplished by passing in a value of 0 (zero) to the barsAgo parameter. Basically, we are saying show me the price data of zero bars ago, which means the current bar.

|  |
| --- |
| **Note**:  In most cases, you will access the historical price series using a core event handler such as OnBarUpdate.  For more advance developers, you may find situations where you wish to access historical price series outside of the core event methods, such as your own custom mouse click.  In these advanced scenarios, you may run into situations where the barsAgo pointer is not in sync with the current bar, and may result in errors when trying to obtain this information.  In those cases, please use the Bars.Get...() methods with the absolute bar index (e.g., [Bars.GetClose(](https://ninjatrader.com/support/helpGuides/nt8/getclose.htm)), [Bars.GetTime()](https://ninjatrader.com/support/helpGuides/nt8/gettime.htm), etc.) |

**Referencing Indicator Data**  
NinjaScript includes a library of built in indicators that you can access. Please see the [Indicator Methods](https://ninjatrader.com/support/helpGuides/nt8/indicators.htm) reference section for clear definitions for how to access each indicator.

All indicator values can be accessed in the following way:

 indicator(parameters)[int barsAgo]

where indicator is the name of the indicator you want to access, parameters is any associated parameters the indicator requires and barsAgo is the number of bars we wish to offset from the current bar.

As an example, we could write a statement to check if the current closing price is greater than the 20 period simple moving average like this:

 Close[0] > SMA(20)[0];

If you wanted to perform the same check but only check against a 20 period simple moving average of high prices you would write it like this:

 Close[0] > SMA(High, 20)[0];

You could write a statement to see if a 14 period CCI indicator is rising like this:

 CCI(14)[0] > CCI(14)[1];

Value of a 10 period CCI 1 bar ago = CCI(10)[1]

Please review the [Indicator Methods](https://ninjatrader.com/support/helpGuides/nt8/indicators.htm) section for proper syntax for accessing different indicator values.

|  |  |
| --- | --- |
| **Navigation:**  [Release Notes](https://ninjatrader.com/support/helpGuides/nt8/release_notes.htm) >  **8.1.2** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/8_1_3.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/release_notes.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/8_1_1.htm) |

See additional patch notes at the bottom

**8.1.2.0 Release Date**

October 25, 2023

|  |
| --- |
| **Features** |
| **SuperDOM Pulling/Stacking column**  Feature #2873    8120PullingStacking    The Pulling/Stacking column is a customizable display that indicates the changes in the market depth based on user settings or if a reset notification is received. For example, in the screenshot above the sell depth at 4520.75 was initially at 91, but dropped to 84 resulting in a display of -7.  Within the columns settings you can change the reset to occur on bid/ask change or when no longer receiving depth data at that level. Additionally, you can adjust reset tolerance and color settings. |
| **SuperDOM Recent column**  Feature #2874    8120Recent    The Recent column is a customizable display that indicates the recent volume that occurred at the bid or ask prices. For example, in the above screenshot we can see a volume of 1 occurred at the bid price and a volume of 4 occurred at the ask.  Within the column's settings you can change the reset to occur when the bid/ask change or when the price returns. Additionally, you can adjust the reset tollerance and color settings. |
| **Selected indicators label displays in bold/italic**  Feature #2894    8120BoldIndicator    Often times multiple indicators can be loaded on a chart and it can be difficult to know what plot is for what indicator and settings. Now you can select the indicator plot and it's label will display in bold and italic to easily and quickly identify it. |
| **Options on Futures data available with NinjaTrader connection**  Feature #1633    8120Options    To match Web Trader, NinjaTrader Desktop can now also display option on futures data with the NinjaTrader connection. |
| **Additional columns available with Option Chain**  Feature #2882    8120OptionsColumns    Additional columns were added to the option chain for additional analysis and to match what's available on Web Trader. |
| **Unlimited number of columns can be added to the Option Chain**  Feature #6010    8120OptionsMultiColumns    Previously only 4 columns could be added to the Option Chain. Now it is possible to enable all available columns to display at the same time. |
| **Chart objects can be sent to back/front**  Feature #197    8120ZOrder    Selecting and right clicking on a chart object now gives the ability to quickly send the object to the front or back of the chart. |
| **Symbology Display Style can be managed within Properties**  Feature #6579    8120Symbology    You can now make Symbology Display Style changes directly within the platform, eliminating the need to access the Client Dashboard for this task. This can be done under Tools> Options> General. |
| **Updated compiler to use Roslyn**  Feature #1642    This enhancement now grants you the flexibility to leverage C# features up to version 8, providing even more versatility in your scripting endeavors. |
| **Workspace loading indicator**  Feature #4451    An animated NinjaTrader icon now displays while workspaces are loading at startup to help reassure everything is in motion and processing smoothly. |

|  |
| --- |
| **Warning**: Some custom scripts that reference old versions of Newtonsoft will need the references updated to Newtonsoft 13 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 6018 | Fixed | Alerts | After an alert sent by a NinjaScript was cleared from the output window it will re-appear in new Alert windows |
| 3341 | Fixed | Backup & Restore | Custom instruments were not saved in backup file |
| 3719 | Fixed | Backup & Restore | Backup reminder prompt did not open the backup window |
| 4438 | Fixed | Backup & Restore | Database was not properly backing up/restoring |
| 247 | Fixed | Chart | Resolved a scenario where the data series window could open unexpectedly |
| 322 | Fixed | Chart | Print chart function wasn't working |
| 4111 | Fixed | Chart | Mini data had display issue with a large list of indicators |
| 7481 | Fixed | Coinbase | Level 2 data stopped working |
| 6560 | Fixed | Continuum/CQG, Orders | Resolved a scenario where an order that was correctly sent got rejected with an incorrect tick value |
| 363 | Fixed | Control Center | Client Dashboard did not load with some set-ups |
| 2253 | Fixed | Control Center | Intermittently the Watch button wasn't detecting if a stream was live |
| 2792 | Fixed | Control Center | WebView Task Bar icon was missing |
| 2897 | Fixed | Control Center | Client Dashboard could require multiple attempts to load |
| 2898 | Fixed | Control Center | Client Dashboard didn't have a header title |
| 2965 | Fixed | Control Center | Exception in Vimeo interactions when no data is received |
| 4590 | Fixed | Control Center | In product announcements were not received if no multi-provider connections were created |
| 4658 | Fixed | Control Center | Gross Realized PnL calculated incorrectly when a lost connection restored |
| 5012 | Fixed | Control Center | WebView instances stayed running after closing NinjaTrader |
| 7161 | Fixed | Control Center | Multiple part fills resulted in unexpected gross realized PnL values |
| 4570 | Fixed | Database | Error occurred with new database if server side ATMs were used |
| 3435 | Fixed | Drawing Tool | Line drawing tool 45 degree snap could place second anchor in wrong location |
| 4636 | Fixed | Drawing Tool | Global drawing objects disappeared from charts when a template was applied |
| 5773 | Fixed | Drawing Tool, Workspaces | Resolved a scenario where a global drawing tool and a multi-data series chart resulted in a lock up |
| 7359 | Fixed | Forex.com | Forex.com demo accounts could not be connected to |
| 174 | Fixed | Forex.com, Orders | Order updates for DFT instruments, which are not supported, were not being ignored |
| 2257 | Fixed | Historical Data Window | Always showed in numerical format |
| 170 | Fixed | Historical Data Window, Charts | Excluded historical data showed in charts and added Changed rows in Historical Data window |
| 6115 | Fixed | Indicator | Indicators were not able to plot on first bar when only one bar of data was loaded |
| 2948 | Fixed | Installer | "Repair" installation ignores custom folder and breaks the installation |
| 209 | Changed | Instruments | Symbology display styles updated for Options on Futures and Index instruments |
| 2258 | Fixed | Instruments | Instruments at times were not rolling over |
| 6819 | Changed | Instruments | Rollover window text updated and link included for more information |
| 3437 | Fixed | Interactive Brokers | Resolved a scenario that prevented some non-US users to not be able to load data |
| 1644 | Fixed | Kintieck, Option Chain | Some expirations didn't show any strike prices |
| 4696 | Changed | Log In | When reCaptcha is needed it now shows immediately and wait time to log in has been reduced to 15 seconds |
| 2868 | Changed | Log In | Added icon to show entered password |
| 1375 | Fixed | Log In | After log in attempts with incorrect credentials, first log in with correct credentials showed an error still |
| 1838 | Changed | Log In | Updated Google sign in authorization |
| 2254 | Fixed | Log In | Recaptcha window was too small when displayed images to verify |
| 4670 | Fixed | Log In | Closing Recaptcha window prevented ability to log in |
| 4961 | Fixed | Log In | Resolved a scenario where log in failed as task canceled |
| 5188 | Fixed | Log In | Style fixes to log in window |
| 4955 | Changed | Market Analyzer, Workspaces | Add SPK to Market Analyzer in default workspaces |
| 2881 | Fixed | Market Analyzer, NinjaScript | Resolved s scenario where a custom script didn't properly display with blank values |
| 320 | Fixed | NinjaScript | Resolved a scenario where a 'Message sent successfully' log wasn't displayed |
| 2265 | Changed | NinjaScript | NTDirect.dll was no longer working. It has been removed and support has been dropped |
| 3429 | Fixed | NinjaScript | Calling a barsAgo index of 1 when CurrentBar is 0 displayed an unexpected bar |
| 3802 | Fixed | NinjaScript | Scripts referencing HLCCalculationMode enum could not be exported/imported as an assembly |
| 2957 | Changed | NinjaScript | Added additional precision to cryptocurrency volume |
| 4729 | Fixed | NinjaScript | AddDataSeries on pre-rollover instrument using custom trading hours failed with incorrect error message |
| 8753 | Changed | NinjaScript | Updated Newtonsoft to version 13 |
| 4287 | Changed | NinjaScript Editor | Only Visual Studio 2019 or 2022  are now supported |
| 4009 | Fixed | NinjaScript Editor | Some system indicators could not properly be copied |
| 3431 | Fixed | NinjaScript, Drawing Tool | There was inconsistency with start/end time errors with @Lines and @Shapes |
| 362 | Fixed | Order | TIF selector was not preventing using of invalid dates |
| 326 | Fixed | Order Flow + | Order Flow Volume Profile drawing tool plotted above price bars if drawn before the first Split date |
| 3805 | Fixed | Order Flow + | There was an incorrect blank cell on Order Flow + Volumetric Bars when using Diagonal Imbalance |
| 4698 | Fixed | Orders | A token not being able to renew prevented order/position updates |
| 4700 | Fixed | Orders | A part filled order that was then canceled still showed as pending cancel |
| 5006 | Fixed | Orders, Server Side ATM | Attempting to use locally simulated orders with server side with server side ATMs didn't show an error |
| 169 | Fixed | Playback | Playing back with historical data and switching instruments caused time to skip |
| 6970 | Changed | Playback | Playback account's settings can now be independently changed from the local simulation account |
| 6552 | Fixed | Playback, Chart | Daily charts displayed the session high/low prior to reaching the levels |
| 5800 | Fixed | Playback, Indicator | Resolved a scenario where an error occurred with the Price Line indicator and switching between Playback and a live connection |
| 329 | Fixed | Playback, NinjaScript | Incorrect prints of bar indexes could occur when disabling and re-enabling strategy via the Strategy tab that was generated in a chart |
| 406 | Changed | Regionalizetion | Buy/sell buttons locations are now regionalized for Korea and can be changed under Options> Trading |
| 704 | Changed | Regionalizetion | Primary and secondary buy/up & sell/down colors have been regionalized for Korea |
| 5408 | Fixed | Server Side ATM | Resolved a scenario where order updates were delayed |
| 298 | Fixed | Share Adapter | Could not send email while using a Hotmail account |
| 1643 | Fixed | Share Adapter | Connect Button for Gmail Share Service is Collapsed in NinjaTrader Desktop |
| 6287 | Changed | Share Adapter | Removed Twitter/X share service |
| 933 | Fixed | Strategies | Orders using isLiveUntilCancelled false couldn't be cancelled with CancelOrder() |
| 2255 | Fixed | Strategy | Copying and pasting a multi-data-series strategy it could change to the secondary series |
| 3427 | Fixed | Strategy | There was inconsistency between sound files played for orders filled via SetStopLoss() and SetProfitTarget() |
| 5426 | Fixed | Strategy | When clicking 'View code' in Strategy Builder, SMA() did not show when assigning it to a custom Series using a different custom series as input for SMA() |
| 6587 | Fixed | Strategy | Resolved a scenario where SetParabolicStop could move an order further away |
| 168 | Fixed | Strategy Analyzer | A Walk Forward optimization with Break At EOD off duplicated results from Friday into the weekend |
| 178 | Fixed | Strategy Analyzer | Some inputs could not be modified after an optimization if Optimize Data Series was set to True |
| 2259 | Fixed | Strategy Analyzer | Genetic optimization with a generation size of 2 got stuck running and did not complete |
| 4052 | Fixed | Strategy Analyzer | Clicking 'View' in the View Strategy column of an AI Generate optimization with an aggregated basket test resulted in an error |
| 4530 | Fixed | Strategy Analyzer | BarsSinceNewTradingDay could be negative 10,000+ when running Genetic Optimization with IsInstantiatedOnEachOptimizationIteration=false |
| 1645 | Fixed | SuperDOM | PnL Freezes when hovering mouse over the Dynamic SuperDOM |
| 6051 | Fixed | TD Ameritrade | Resolved an 'object reference not set to an instance of an object' error when connecting |
| 4641 | Fixed | Tick Replay, Indicator | Swing indicator OnPriceChange showed different results than OnBarClose when Tick Replay was enabled |
| 5345 | Fixed | Trade Performance | Some regions/timezone settings prevented reports from running |
| 5421 | Fixed | Trade Performance | In some scenarios trades could show with the wrong timezone |
| 5568 | Fixed | Trade Performance | PC Region settings set to Poland prevented reports from running |
| 6026 | Fixed | Trade Performance | Resolved a scenario where historical MIT orders caused an error |
| 6107 | Fixed | Trade Performance | Resetting a template did not reset AI Generate properties |
| 2246 | Changed | Workspaces | Removed Eurex instruments from the default workspaces |

**8.1.2.1 Release Date**

November 27, 2023

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 11099 | Fixed | NinjaTrader Connection | Resolved a scenario that could cause unexpected failed log ins applied to an account |
| 10687 | Fixed | NinjaScript | Scripts could incorrectly be exported with versions of Agile older than 6.9.1.2 |
| 10964 | Fixed | NinjaScript Output Window | Window didn't properly auto scroll with new prints |
| 11378 | Fixed | NinjaScript | Reference was missing to Microsoft.Csharp.dll |
| 10997 | Fixed | NinjaScript Editor | Intellisense could show code for unsupported versions of C# |
| 11165 | Fixed | Simulation | An exception occurred when creating a sim account |
| 10969 | Fixed | Skins | Custom PnL colors were not being applied |
| 11037 | Fixed | Indicators | McClellan Oscillator wouldn't load with some symbology display styles |
| 9931 | Fixed | Rithmic | Resolved a scenario that resulted in an unknown instrument |
| **Navigation:**  [Operations](https://ninjatrader.com/support/helpGuides/nt8/operations.htm) > [Charts](https://ninjatrader.com/support/helpGuides/nt8/charts.htm) >  **Working with Indicators** | | | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/chart_styles.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/charts.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/working_with_drawing_tools__ob.htm) |

NinjaTrader comes with over 100 pre-built technical indicators, which can be added, removed and edited via the Indicators window. Indicators can be applied to [charts](https://ninjatrader.com/support/helpGuides/nt8/charts.htm), the [SuperDOM](https://ninjatrader.com/support/helpGuides/nt8/superdom.htm), or [Market Analyzer](https://ninjatrader.com/support/helpGuides/nt8/working_with_columns.htm) columns, and custom technical indicators can be created via the [NinjaScript Editor](https://ninjatrader.com/support/helpGuides/nt8/editor.htm).

| playVideo |
| --- |
|  |

tog_minus        [Understanding the Indicators window](javascript:HMToggle('toggle','UnderstandingTheIndicatorsWindow','UnderstandingTheIndicatorsWindow_ICON'))

|  |
| --- |
| The **Indicators** window is used to add, remove and edit all indicators within a chart.    **Accessing the Indicators Window from a Chart**  There are multiple ways to access the **Indicators** window from a chart:    •Left click on the **Indicators** icon in the chart toolbar  •Right mouse click in the chart background when no chart object is selected, and select the **Indicators** menu item  •Double click on an indicator within a chart  •Right click on a highlighted indicator within a chart and select the **Properties** menu item  •Use the default Ctrl + I [Hot Key](https://ninjatrader.com/support/helpGuides/nt8/hot_key_manager.htm) when the chart has focus.    **Sections of the Indicators Window**  The image below displays the three sections of the **Indicators** window.    Indicators1    1.The "Available" section displays a list of available indicators  2.The "Configured" section displays indicators currently applied to the chart or SuperDOM  3.The "Properties" section displays the selected indicator's parameters |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators.htm#UnderstandingTheIndicatorsWindow)

tog_minus        [How to add an Indicator](javascript:HMToggle('toggle','HowToAddAnIndicator','HowToAddAnIndicator_ICON'))

|  |
| --- |
| **Adding an Indicator**  To add an indicator to a chart:    1.Open the **Indicators** window *(see the "Understanding the Indicators window" section above)*  2.Left mouse click on the indicator you want to add in the "Available" section, then press the **add** option in the "Configured" section. Alternatively, you can simply double click on the indicator in the "Available" section to add it to the "Configured" section.  3.The indicator will now be visible in the "Configured" section  4.The indicator's parameters will now be editable on the right side of the **Indicators** window *(see the "How to edit an indicator" section below)* |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators.htm#HowToAddAnIndicator)

tog_minus        [How to edit an Indicator's parameters](javascript:HMToggle('toggle','HowToEditAnIndicatorsParameters','HowToEditAnIndicatorsParameters_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Editing an Indicator**  You can customize any indicator from the **Indicators** window.    1.Open the **Indicators** window (see the*"Understanding the Indicators window"*section above)  2.Highlight the indicator you would like to edit from the list of applied indicators  3.Once highlighted, this indicator's parameters will be available to edit in the "Properties" section.    **Chart Indicator Parameters**  The following parameters are common to all indicators applied on a chart:    Indicators2     |  |  | | --- | --- | | Input Series | Please see the "*Indicator Input Series*" section on this page for further information. | | Calculate | Sets the frequency at which the indicator performs its calculations. See the note below for information on each possible setting for this property | | Label | The label displayed on the chart. Leaving the field blank will remove the label from being displayed on the chart. Enclosing a label in quotations ("MyEMA" for example) will display the text within the quotations and exclude the system added trailing series information. | | Maximum Bars Look Back | Determines the maximum number of bars the indicator can look back to perform calculations on historical data. This is set to 256 by default (the most memory-friendly setting), but it can be changed to "infinite" to allow for a greater look back period. | | Auto Scale | When enabled, the indicator will be included in the chart panel's vertical automatic scaling | | Displacement | Sets the number of bars by which to displace the indicator plots | | Display in Data Box | Enables or disables the inclusion of the indicator's plot values in the **Data Box** | | Panel | Sets the panel in which the indicator is plotted. If you select "Same as input series," the indicator will be linked to the **Input Series** and automatically move if the **Input Series** is modified to a different panel. | | Price marker(s) | When enabled, the indicator value is plotted in the axis selected under the "Scale Justification" property. | | Scale justification | Sets the scale on which the indicator will be plotted. Possible values are "Right," "Left," and "Overlay" | | Visible | Enables or disables visibility and function of the indicator on the chart | | Plots | Sets a variety of parameters, such as color, for the plots drawn by the indicator |      |  | | --- | | **Note**: The "Calculate" property offers three possible settings to control how often an indicator performs its calculations:  •On Bar Close - Run calculations once on the close of each bar of the **Input Series**  •On Each Tick - Run calculations on each incoming tick of price data (CPU intensive)  •On Price Change - Runs calculations on each change in price |     **Saving an Indicator's Parameters**  You can optionally save your customized indicator's parameters as templates. Saving it as Default will recall your customized settings the next time you add that specific indicator to a chart.    Please see the [Saving Chart Defaults and Templates](https://ninjatrader.com/support/helpGuides/nt8/saving_chart_defaults_and_templates.htm) page for more information.    **Indicator Input Series**  The indicator **Input Series** dialogue allows you to select the **Input Series** for your indicator's calculations. To access this window, left mouse click within the "Input Series" field. You can then select the Close, High, Low, Median, Open, Typical, or Weighted price of any **Data Series** applied to the chart. Alternatively, you can choose another indicator as the input series. When you select another indicator as the input series, The "Properties" section of the **Input Series** dialogue will display properties related to the indicator being used as the **Input Series**, allowing you to configure it to your desired settings. This allows you to nest multiple indicators. Once you have selected the **Input Series** of your choice, left mouse click the **OK** button to exit the **Input Series** window.    Indicators4    In the image above, we can select one of the **Data Series** applied to the chart, or another indicator, for use as an indicator's**Input Series**.     |  | | --- | | **Note**: To take advantage of this feature NinjaScript indicators will need to implement the [Input](https://ninjatrader.com/support/helpGuides/nt8/input.htm) ISeries as their main data input. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators.htm#HowToEditAnIndicatorsParameters)

tog_minus        [How to remove an Indicator](javascript:HMToggle('toggle','HowToRemoveAnIndicator','HowToRemoveAnIndicator_ICON'))

|  |
| --- |
| **Removing an Indicator From a Chart**  There are three ways to remove an indicator from a NinjaTrader chart:    •Open the Indicators window (see the *"Understanding the Indicators window"*section above). Next, select an indicator from the "Configured" section, then select the **Remove** option, and finally press the **OK** button to exit the **Indicators** window.  •Left mouse click to select the indicator on your chart, then press the Delete key on your keyboard.  •Left mouse click to select the indicator on your chart, then right mouse click the indicator and select the **Remove** menu item. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators.htm#HowToRemoveAnIndicator)

tog_minus        [Custom Indicator development](javascript:HMToggle('toggle','CustomIndicatorDevelopment','CustomIndicatorDevelopment_ICON'))

|  |
| --- |
| In addition to the indicators that come pre-built with the NinjaTrader application, you also have the ability to create custom indicators of your own. For example, you could create your own custom multi-series indicators using price and volume data to apply to your charts or share with fellow traders.    For more information on using NinjaScript to build custom indicators please see the [NinjaScript section](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) of the user help guide, or click [here](https://ninjatrader.com/support/helpGuides/nt8/indicator.htm) to view NinjaScript indicator-development tutorials. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators.htm#CustomIndicatorDevelopment)

tog_minus        [Working with Indicators in Market Analyzer columns](javascript:HMToggle('toggle','WorkingWithIndicatorsInMarketAnalyzerColumns','WorkingWithIndicatorsInMarketAnalyzerColumns_ICON'))

|  |
| --- |
| Please see the [Working With Columns](https://ninjatrader.com/support/helpGuides/nt8/working_with_columns.htm) page for information on working with indicators in **Market Analyzer** columns. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators.htm#WorkingWithIndicatorsInMarketAnalyzerColumns)

tog_minus        [Working with Indicators in the SuperDOM](javascript:HMToggle('toggle','WorkingWithIndicatorsInTheSuperDOM','WorkingWithIndicatorsInTheSuperDOM_ICON'))

|  |
| --- |
| Please see the**SuperDOM** [Working with Indicators](https://ninjatrader.com/support/helpGuides/nt8/working_with_indicators_superdom.htm) page for information on working with indicators in the **SuperDOM**. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators.htm#WorkingWithIndicatorsInTheSuperDOM)

|  |  |
| --- | --- |
| **Navigation:**  [Operations](https://ninjatrader.com/support/helpGuides/nt8/operations.htm) > [Charts](https://ninjatrader.com/support/helpGuides/nt8/charts.htm) > [Order Flow +](https://ninjatrader.com/support/helpGuides/nt8/order_flow_plus.htm) >  **Order Flow Volumetric Bars** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/order_flow_volume_profile.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/order_flow_plus.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/order_flow_vwap.htm) |

NinjaTrader Order Flow Volumetric [bars](https://ninjatrader.com/support/helpGuides/nt8/bar_types.htm) provide a detailed ‘x-ray’ view into each price bar’s aggressive buying and selling activity. This technique primarily attempts to answer the question which side was the most aggressive at each price level. This is done by calculating the delta (greek for difference) between buying and selling volume (please see the Delta type property explanation below).

With the delta value known for each price level in the bar, it is then classified per each session for analysis and emphasizes the buying / selling strength unfolding. This is done by a gradient coloring approach shading the value cells in the bar, where the level of sensitivity for the gradient can be set via the Shading sensitivity property. The higher this value is set, the finer the gradient can be applied to various levels of strength - the NinjaTrader default is 20 levels.

This can be thought of as a way of not only saying who ‘won / lost’ the price level’s auction, but also by what margin or strength. This is not a signal in itself per se, but rather a mechanical means to classify the buying vs selling activity at each individual price level and thus offer the trader a more detailed look what happens inside the price bars.

A second comparison of buy sell volumes is the Imbalance detection. Here the price level buying and selling volumes are compared diagonally to understand which side of the market was stronger by exceeding the set Imbalance ratio. For example if the buying volume was 1000 contracts and the selling volume diagonal below was 300 then buying Imbalance was detected (assuming a default Imbalance ratio of 1.5). This can be helpful especially if multiple Imbalances 'cluster' close together to form support / resistance areas.

NinjaTrader Order Flow Volumetric bars can provide a large degree of details and facilitate displaying the information in a dynamically sized way, as the text is re-sized as your horizontally or vertically adjust the chart's scale range.

|  |
| --- |
| **Critical**: To perform the delta buy / sell aggressor classification (DeltaType BidAsk), historical bid / ask tick data access by your provider is needed. To see which data providers can offer which type of data in NinjaTrader, please review [this table](https://ninjatrader.com/support/helpGuides/nt8/data_by_provider.htm). If your provider could only support 'last' historical tick data, then the classification could still be made using DeltaType UpDownTick mode.    Forex spot data charting is traditionally driven from the bid side only, since no 'true' last exists. Provided values would not represent a centralized auction based market as with stocks or futures - as such Order Flow Volumetric Bars would not be supported on Forex spot data. |

tog_minus        [Order Flow Volumetric Overview](javascript:HMToggle('toggle','OrderFlowVolumetricOverview','OrderFlowVolumetricOverview_ICON'))

|  |
| --- |
| **Order Flow Volumetric Overview:**    You can apply the Order Flow Volumetric bars within a Chart Data Series window under **Type**.    OrderFlowVolumetric    Below we show a 5 minute Order Flow Volumetric BidAsk style chart of the popular E-Mini S&P 500 contract. An exemplary bar in yellow is annotated to show the different components you will work with on an Order Flow Volumetric chart.    1. Order Flow Volumetric bar  2. Sell Volume per each price level seen in the bar  3. Buy Volume per each price level seen in the bar  4. Open / Close bar  5. Maximum highlight in the bar - this shows the price(s) with the highest volume or delta in the bar  6. Bar Statistics panel (only 3 out of possible 10 values activated here)    OFVM1    On the next image, we see an excerpt of the same chart, however now the chart style type is changed to Delta. The bar highlighted yellow corresponds to the annotated bar above - this style, instead of the individual buy / sell volumes, show the combined delta value for each price level.  If positive the level was seeing buying strength, negative if selling strength.    OFVM2    Now let's take a look how the actual values we see in the bar are calculated from the market data, as an example take the first level of our Volumetric BidAsk bar in the first screen-shot (numbers right below the 2/3 annotated numbers) :    We see a bid or selling volume of **220** and a ask or buying volume of **740**.    Taking the difference (Buy volume - Sell volume) so **740 - 220** we get the delta value of **520** - which we see as first cell value in the 2nd screen-shot showing the Delta type.    The shading taking place in the bars will always be based on this delta value calculated, the chart style just defines what kind of textual data detail will be displayed (the actual Bid Ask Volumes or the Delta value).    Moving on to the lower portion Bar Statistics (6), we see the Buy Vol and Sell Vol for the bar and a delta as well exposed - however, please note that these are summed values for the entire bar.    If we sum all the buy volumes we would get **3463**; summing sell volumes for the bar is **4535**. All the price level deltas summed would equal the delta of the entire bar **-1072**.    **Order Flow Volumetric Imbalance charting:**    The screen-shot below explains the workings of the Imbalance detection in more detail. You can see the Buy / Sell volumes are compared diagonally here to arrive at the classification if buy or sell imbalance is present.    Let's run through the first calculation for the example bar in yellow:    1. Buy volume of **518** is compared to the diagonal below sell volume of **989**  2. Dividing the buy volume into the sell volume we get a ratio of **989 / 518 = 1.9092...**  3. NinjaTrader by default sets the ratio for Imbalance at 1.5, so this level gets marked with Sell Imbalance (magenta text color per default).  4. As a further condition, a minimum difference between the compared values must be present. This value is defaulted to 10 - which is valid in our example as well.    In the case that both Imbalance and Maximum would trigger for the same cell, the Maximum would override and be displayed (example shown below at annotation 1).    NinjaTrader 8.0.19.0 or newer also offers the option to compare horizontal for Imbalance (Imbalance mode setting).    OFVM3    **Order Flow Volumetric Bar Statistics:**    The Volumetric Bar statistics show important values for each Volumetric bar in a static grid-like fashion. The same gradient strength shading as for the main Volumetric bars is applied here.    1.Via a right click in the price scale section the individual statistic values could be enabled / disabled 'on the fly'.    OFVM4    **Order Flow Volumetric Bar data shown as profile:**    Below chart is an example of showing the volumetric bid ask volume bar data as distribution profile, additionally 'hide text' is checked - which means we see the maximum (yellow) as well as imbalance (cyan / magenta) marked via the cell borders coloring.    Showing the data in this fashion can give traders an easier read, as differences between light and high volume price areas becomes visually more striking.    OFVM7 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order_flow_volumetric_bars.htm#OrderFlowVolumetricOverview)

tog_minus        [Order Flow Volumetric Imbalance Customization Example](javascript:HMToggle('toggle','OrderFlowVolumetricImbalanceCustomizationExample','OrderFlowVolumetricImbalanceCustomizationExample_ICON'))

|  |
| --- |
| **Order Flow Volumetric Imbalance Customization example:**    This section presents an example of how NinjaTrader Order Flow Volumetric bars can be highly customized to your trading style. Traders focused on Order Flow Volumetric Imbalances may consider working these charting ideas into their NinjaTrader setup.    The settings we present below could be used as a starting point -    OFVM6    The regular 5 Minute CandleStick chart is brought in to this chart via a second Data Series, so forming a [MultiSeries chart](https://ninjatrader.com/support/helpGuides/nt8/working_with_multiple_data_series.htm) with our main 5min Order Flow Volumetric bars. This can advantageous if you prefer to plot the regular bar / candlestick portion in the middle of the bar between the Buy/Sell volume columns. In NinjaTrader 8.0.19.0 or newer, this can now be also accomplished without a second DataSeries by enabling the Center Open/Close bar plotting option.    OFVM5 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order_flow_volumetric_bars.htm#OrderFlowVolumetricImbalanceCustomizationExample)

tog_minus        [Order Flow Volumetric Bars parameters](javascript:HMToggle('toggle','OrderFlowVolumetricBarsDataSeries','OrderFlowVolumetricBarsDataSeries_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Data series:     |  |  | | --- | --- | | Base period type | Sets the base period type the Volumetric bars should be calculated on, possible values include:    Tick,  Volume,  Second,  Minute,  Day,  Week,  Month,  Year,  Range | | Base period value | Period of your chosen Base period type for Volumetric bars (i.e. **3** Minute, **450** Tick) | | Delta type | Sets how the delta is calculated for buy / sell aggressor classification. Possible values are:  BidAsk or UpDownTick.    BidAsk - Accumulates the volume of orders filled at the bid or less vs ask or more. Orders filled at the ask or more price are considered buying pressure. Orders filled at the bid or less price are considered selling pressure. If the current tick price is between ask and bid, the volume will be recorded to the same pressure as the previous tick.    UpDownTick - Accumulates the volume of up ticks vs down ticks. Up ticks are considered buying pressure. Down ticks are considered selling pressure. If the current tick price is the same as the previous tick price, the volume will be recorded to the same pressure as the previous tick. | | Ticks per level | Sets the level of aggregation for individual price levels, i.e. if price levels should be merged together, default 1 – so each price level delta result is seen individually inside the price bars    (Please note that with a higher Ticks per level set, there could be Volumetric remainder cells as a result that are actually smaller than your set Ticks per level, as not all bar ranges could be evenly divisible by the Ticks per level value) | | Size filter | Default 0, could be set higher to limit seeing the delta only for trades higher than the size filter setting, for example tracking larger trades only - keep in mind this could be potentially set too high for your chart, so Volumetric bars value cells could show 0 volume, so it never met this criteria then to be considered in the analysis. | | Tick replay | Check to allow indicators or strategies to access Tick replay data, the main Volumetric bars though will always be built off a 1 tick series. |     Chart style:     |  |  | | --- | --- | | Chart style type | Sets the chart style type used, possible values are : BidAsk or Delta    BidAsk - allows you seeing the individual buy / sell volume cells inside the Volumetric bar, buy (ask) volume is shown on the right side, sell (bid) volume shown on the right side. Additionally this style can highlight the maximum and imbalance conditions.    Delta - allows to see the single value total net delta per each price cell, maximum can be shown as well (no imbalance option for this style) | | Center open close bar | When checked, allows centering of the Open / Close bar on the BidAsk Chart style type | | Show as profile | Enables to show the Volumetric bar data as profile / distribution | | Show volume | Enables to show the volume distribution to the right of the Volumetric bar. | | Strength sensitivity | Sets how many gradient levels should be calculated to provide the buy / sell strength shading for the Volumetric bars and Bar Statistics, the default setting is 20. This is reset at every session break to ensure the gradient strength classifications easily comparable across various days market action. | | Size display filter | Allows only displaying Bid / Ask or delta numbers higher than the threshold. Default 0 and a visual only setting. | | Show imbalance | Enables the display of Imbalance coloring    Only applicable for the Volumetric BidAsk chart style type    With 'Hide text' checked will display imbalances by coloring the respective imbalance cell borders | | Imbalance ratio | Sets the ratio used for comparing the buy / sell volumes for accessing if Imbalance is present, default 1.5    Only applicable for the Volumetric BidAsk chart style type | | Imbalance mode | Sets the comparison mode for Imbalance : Diagonal or Horizontal | | Minimum delta for imbalance | Sets the minimum delta to be seen diagonally across compared buy / sell volume columns for displaying Imbalance, default value is 10.    Only applicable for the Volumetric BidAsk chart style type | | Color dominant side | Enables to display the dominant ('winning') strength side only in the Volumetric strength display    Only applicable for the Volumetric BidAsk chart style type | | Box outline | Sets options for the display of the outline of the Volumetric boxes | | Box grid | Sets options for the display of the inner grid of the Volumetric bars | | Candle body outline | Sets options for the display of the body outline of the Open/Close bar | | Color for up bars | Sets the up brush color used for Open/Close bar | | Color for down bars | Sets the down brush color used for Open/Close bar | | Color for doji | Sets the doji brush color used for Open/Close bar | | Color for positive strength | Sets the positive brush color used for positive strength gradients | | Color for negative strength | Sets the negative brush color used for negative strength gradients | | Color for buy imbalance | Sets the brush color used for buy imbalance    Only applicable for the Volumetric BidAsk chart style type | | Color for sell imbalance | Sets the brush color used for sell imbalance    Only applicable for the Volumetric BidAsk chart style type | | Show maximum | Enables the display of the maximum value for the Volumetric bars, if identical values would be seen across cells, then all cells sharing the maximum would be highlighted.    BidAsk style: If Color dominant side is checked, it will highlight both the Buy and Sell price levels with the highest volume, else it will highlight the price level with the highest combined Buy / Sell volume (Total volume for the price level)    Delta stye: highest absolute delta level would be highlighted    With 'Hide text' checked will display the maximum by coloring the respective maximum cell borders | | Color for maximum | Sets the color used for maximum display | | Hide text | Enables to hide the Volumetric cell values text display | | Color for text | Sets the text color used for the displaying the Volumetric cell values | | Show bar statistics | Enables the display of the Volumetric bar statistics in the lower portion (static location) of the chart. You can checkmark each desired statistic, possible statistics are -     |  |  | | --- | --- | | Trades | The total number of trades in the bar | | Volume | The volume for the bar | | Buy vol | The Buy vol for the bar | | Sell vol | The Sell vol for the bar | | Delta (bar) | The total delta for the bar (all individual price level delta summed) | | Delta (%) | The bar delta expressed as percentage of volume for the bar | | Cumulative delta | The bar delta value cumulated throughout the session | | Min delta | The minimum delta seen in the bar (intrabar). This could be positive as well, i.e. a strong up bar with no selling pressure | | Max delta | The maximum delta seen in the bar (intrabar). This could be negative as well, i.e. a strong down bar with no buying pressure | | Delta change | The change in delta from the previous bar's delta value | | Delta SH | The delta since last time price touched the high of the bar, usually negative | | Delta SL | The delta since last time price touched the low of the bar, usually positive. |     The statistics can be toggled as well 'on the fly' from the chart's right scale by clicking on the statistic labels. | | Color for base | Sets the brush color used for base bar statistics strength gradients (Trades, Volume) | | Statistics grid | Sets options for the display of the bar statistics grid of the Volumetric bars | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order_flow_volumetric_bars.htm#OrderFlowVolumetricBarsDataSeries)

tog_minus        [Order Flow Volumetric Values NinjaScript access](javascript:HMToggle('toggle','OrderFlowVolumetricValuesNinjaScriptAccess','OrderFlowVolumetricValuesNinjaScriptAccess_ICON'))

|  |
| --- |
| For information on how to access the Order Flow Volumetric Bars and Bar Statistic values in NinjaScript, please see the [Order Flow Volumetric Bars](https://developer.ninjatrader.com/docs/desktop/order_flow_volumetric_bars) page in the NinjaScript section of the Help Guide. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order_flow_volumetric_bars.htm#OrderFlowVolumetricValuesNinjaScriptAccess)

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/support/helpGuides/nt8/strategy.htm) > [Order Methods](https://ninjatrader.com/support/helpGuides/nt8/order_methods.htm) > [Managed Approach](https://ninjatrader.com/support/helpGuides/nt8/managed_approach.htm) >  **Advanced Order Handling** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/managed_approach.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/managed_approach.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/managed_cancelorder.htm) |

Advanced order handling is reserved for **EXPERIENCED** programmers. Through advanced order handling you can submit, change and cancel orders at your discretion through any event-driven method within a strategy. Each order method within the "Managed Approach" section has a method overload designed for advanced handling.

tog_minus        [Live Until Cancelled Orders](javascript:HMToggle('toggle','LiveUntilCancelledOrders','LiveUntilCancelledOrders_ICON'))

|  |
| --- |
| Orders can remain live until you call the [CancelOrder()](https://ninjatrader.com/support/helpGuides/nt8/managed_cancelorder.htm) method, or until the order's time in force has expired, whichever comes first. This flexibility allows you to control exactly when an order should be cancelled instead of relying on the close of a bar. Each order method, such as [EnterLongLimit()](https://ninjatrader.com/support/helpGuides/nt8/enterlonglimit.htm), has a method overload designed to submit a "live until canceled" order. When using this overload, it is important to retain a reference to the Order object, so that it can be canceled via CancelOrder() at a later time. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?advanced_order_handling.htm#LiveUntilCancelledOrders)

tog_minus        [The Order Class](javascript:HMToggle('toggle','TheOrderClass','TheOrderClass_ICON'))

|  |  |  |
| --- | --- | --- |
| All order methods return an [Order](https://ninjatrader.com/support/helpGuides/nt8/order.htm) object. There are several important items to note:    •An Order object returned from calling an order method contains dynamic properties which will always reflect the current state of the associated order  •The property <Order>.OrderId is **NOT** a unique value, since it can change throughout an order's lifetime.  Please see the section below on "*Transitioning order references from historical to live"* for details on how to handle.  •To check for equality, you can compare Order objects directly    The following example code demonstrates the submission of an order and the assignment of the Order return object to the variable "entryOrder." After this, the object is checked in the OnOrderUpdate() method for equality, and then checked for the Filled state.    **Examples**   | ns | | --- | | private Order entryOrder = null; protected override void OnBarUpdate() {     if (entryOrder == null && Close[0] > Open[0])         EnterLong("myEntryOrder"); }   protected override void OnOrderUpdate(Order order, double limitPrice, double stopPrice, int quantity, int filled, double averageFillPrice, OrderState orderState, DateTime time, ErrorCode error, string nativeError) {  *// Assign entryOrder in OnOrderUpdate() to ensure the assignment occurs when expected.*     *// This is more reliable than assigning Order objects in OnBarUpdate, as the assignment is not gauranteed to be complete if it is referenced immediately after submitting*     if (order.Name == "myEntryOrder" && orderState != OrderState.Filled)       entryOrder = order;    // Null Entry order if filled or cancelled. We do not use the Order objects after the order is filled, so we can null it here    if (entryOrder != null && entryOrder == order)     {        if (order.OrderState == OrderState.Cancelled && order.Filled == 0)            entryOrder = null;        if (order.OrderState == OrderState.Filled)            entryOrder = null;     } } | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?advanced_order_handling.htm#TheOrderClass)

tog_minus        [Transitioning order references from historical to live](javascript:HMToggle('toggle','Transitioningorderreferencesfromhistoricaltolive','Transitioningorderreferencesfromhistoricaltolive_ICON'))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| When starting a strategy on real-time data, the [starting behavior](https://ninjatrader.com/support/helpGuides/nt8/syncing_account_positions.htm) will renew any active historical orders and resubmit these orders to your live or simulation account.  This process includes updating the historical/backtest generated order ID to the account generated order ID, and any associated OCO IDs.  If you are tracking order objects, is critical that you update the order reference to ensure that it is now using the correct order details.    This should be done in [OnOrderUpdate()](https://ninjatrader.com/support/helpGuides/nt8/onorderupdate.htm) to ensure all cases of order transitions are handled.     |  | | --- | | **Critical**:  If you **DO NOT** update a historical order reference, and then attempt to cancel/change that order *after* it has been submitted in real-time, your strategy will be disabled with a message similar to: *"Strategy has been disabled because it attempted to modify a historical order that has transitioned to a live order."* |        |  | | --- | | **Tip**:  When the real-time order is submitted, there is a generic Order object passed into the [OnOrderUpdate()](https://ninjatrader.com/support/helpGuides/nt8/onorderupdate.htm) method containing the live order details which can be used for debugging.  It is recommended you use the helper [GetRealtimeOrder()](https://ninjatrader.com/support/helpGuides/nt8/getrealtimeorder.htm) when your strategy transitions to real-time to update your order references |       **Example**   | ns | | --- | | private Order entryOrder = null;    protected override void OnBarUpdate()  {     if (entryOrder == null && Close[0] > Open[0])        entryOrder = EnterLongLimit("myEntryOrder", Low[0]);  }  protected override void OnOrderUpdate(Order order, double limitPrice, double stopPrice, int quantity, int filled, double averageFillPrice, OrderState orderState, DateTime time, ErrorCode error, string nativeError)  {    // One time only, as we transition from historical    // Convert any old historical order object references to the live order submitted to the real-time account    if (entryOrder != null && entryOrder.IsBacktestOrder && State == State.Realtime)        entryOrder = GetRealtimeOrder(entryOrder);       // Null entryOrder if filled or cancelled. We do not use the Order objects after the order is filled, so we can null it here    if (entryOrder != null && entryOrder == order)     {        if (order.OrderState == OrderState.Cancelled && order.Filled == 0)            entryOrder = null;        if (order.OrderState == OrderState.Filled)            entryOrder = null;     }  } | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?advanced_order_handling.htm#Transitioningorderreferencesfromhistoricaltolive)

tog_minus        [Working with a Multi-Instrument Strategy](javascript:HMToggle('toggle','WorkingWithAMultiinstrumentStrategy','WorkingWithAMultiinstrumentStrategy_ICON'))

|  |  |  |
| --- | --- | --- |
| With advanced order handling, you can submit an order in the context of any [Bars](https://ninjatrader.com/support/helpGuides/nt8/bars.htm) object by designating the "BarsInProgress" index. For example, if your primary bar series is "MSFT" and your secondary series added to the strategy through the AddDataSeries() method is "AAPL", you can submit an order for either "MSFT" or "AAPL" from anywhere within the strategy. In addition to the information found in the [multi-time frame and instrument strategies](https://ninjatrader.com/support/helpGuides/nt8/multi-time_frame__instruments.htm) page, this section specifically covers order submission.    As an example, consider the EnterLongLimit() method and one of its method overloads designed for advanced order handling:      EnterLongLimit(int barsInProgressIndex, bool isLiveUntilCancelled, int quantity, double limitPrice, string signalName)    In this example, an "MSFT 1 minute" chart is the primary bar series on which the strategy is running. A secondary bar series is added for "AAPL 1 minute" via the AddDataSeries() method in the [OnStateChange()](https://ninjatrader.com/support/helpGuides/nt8/onstatechange.htm) event handler. After adding the secondary Bars object, MSFT has a [BarsInProgress](https://ninjatrader.com/support/helpGuides/nt8/barsinprogress.htm) index of 0 and AAPL has an index value of 1.    The following example code demonstrates how to monitor for bar update events on the first instrument, while submitting orders to the second instrument.    **Example**   | ns | | --- | | private Order entryOrder = null;    protected override void OnStateChange() {   if (State == State.Configure)   {       AddDataSeries("AAPL", BarsPeriodType.Minute, 1);   } }  protected override void OnBarUpdate() {     // Check if the MSFT series triggered an bar update event     if (BarsInProgress == 0)     {         // Submit an order for AAPL in the context of MSFT bar update event         if (entryOrder == null)               EnterLongLimit(1, true, 1, Lows[1][0], "AAPL Order");     } }    protected override void OnOrderUpdate(Order order, double limitPrice, double stopPrice, int quantity, int filled, double averageFillPrice, OrderState orderState, DateTime time, ErrorCode error, string nativeError) {     // Assign entryOrder in OnOrderUpdate() to ensure the assignment occurs when expected.     // This is more reliable than assigning Order objects in OnBarUpdate, as the assignment is not gauranteed to be complete if it is referenced immediately after submitting     if (order.Name == "AAPL Order" && orderState != OrderState.Filled)       entryOrder = order; } | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?advanced_order_handling.htm#WorkingWithAMultiinstrumentStrategy)

|  |  |
| --- | --- |
| **Navigation:**  [Release Notes](https://ninjatrader.com/support/helpGuides/nt8/release_notes.htm) > [8.0](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) >  **8.0.15.1** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/8_0_16_3.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/8_0_14_2.htm) |

**8.0.15.0 Release Date**

July 30, 2018

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature #** | **Status** | **Category** | **Comments** |
| 12351 | Added | Order Flow + | Added Order Flow Volume Profile indicator |
| 12381 | Added | Order Flow + | Added Order Flow Volume Profile drawing tool |
| 12916 | Added | Coinbase | Added Cryptocurrency support via Coinbase (See the Coinbase connection guide [here](https://ninjatrader.com/ConnectionGuides/Coinbase-Connection-Guide)) |
| 12798 | Added | Drawing | Added Polygon drawing tool |
| 12873 | Added | Drawing | Added Time Cycles drawing tool |
| 12777 | Added | Indicator | Added McClellan Oscillator indicator |
| 12818 | Added | Indicator | Added Relative Vigor Index indicator |
| 12909 | Added | Indicator | Added Wiseman Awesome Oscillator indicator |
| 12910 | Added | Indicator | Added Wiseman Alligator indicator |
| 12911 | Added | Indicator | Added Wiseman Fractal indicator |
| 12871 | Added | NinjaScript Editor | Added an animated icon to show when compiling in the NinjaScript editor |
| 12882 | Added | NinjaScript | Added overload for NinjaScript: AddDataSeries(string instrumentName), allowing same Bars Period with a different instrument |
| 13014 | Added | Strategy | Added Parabolic Stop support for NinjaScript |
| 12937 | Added | Strategy Analyzer | Added Max Strength optimization metric |
| 12967 | Added | Strategy Analyzer | Added Probability statistic to the summary tab |
| 12305 | Added | Workspaces | Added automatic backup of saved workspaces which can be restored under Tools > Database Management |
| 12938 | Added | Chart, Indicator | Added "Calculating..." label for when indicators are still processing historical data |
| 12942 | Added | Historical Data | Added data importer for tickdata.com |

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 12722 | Changed | Alerts | Improved Alert Pop Up message formatting |
| 12855 | Fixed | Alerts | Resolved scenario where alerts using global drawing objects no longer functioned after a restart |
| 12884 | Fixed | Alerts | Removed invalid drawing objects pop up that could occur when rolling over |
| 12921 | Fixed | Alerts | Reloading NinjaScript in a chart resulted in alerts tied to indicators or drawing objects to stop working |
| 12977 | Fixed | Alerts | Exporting from Alerts log to Excel caused the time to be exported incorrectly |
| 12854 | Fixed | ATM Strategies | Resolved some scenarios where adjusting settings could result in a TIF error |
| 12970 | Fixed | ATM Strategies | Modified stop value was not respected when scaling in with an ATM that uses simulated stops |
| 12952 | Fixed | BarType | Heiken Ashi based on tick bars created different, rather than modified, bars |
| 12666 | Fixed | Chart | Increasing a chart's days to load did not function if it was increased by less than 4 days |
| 12794 | Fixed | Chart | In some scenarios moving an indicator to a different panel could leave an empty panel |
| 12861 | Fixed | Chart | Price marker disappeared on non-equidistant chart when more granular series reached right edge of screen |
| 12869 | Fixed | Chart | When equidistant was set to false multi-series charts with time-based and daily series used wrong series for latest bar |
| 12878 | Fixed | Chart | Chart bars refresh could result in incorrect panel values |
| 12975 | Fixed | Chart | Incorrect dates were shown with global cross-hairs when the cursor was along the x-axis |
| 12860 | Fixed | Chart | Orders on instruments that display values as fractions did not display order as a fraction when modifying |
| 12979 | Fixed | CQG, Continuum, Connections | Resolved a case where some order states prevented a connection |
| 12968 | Fixed | Data | Metastock MASTER file was not recognized by File Explorer |
| 12974 | Fixed | Data | There was no pop up error when attempting to import Metastock data using the 64-bit version |
| 12972 | Fixed | Drawing | When snapping Trend Channel to price it would move in tick increments away from copied price instead of snapping to tick boundaries |
| 12863 | Fixed | DrawingTool | Moving draw objects could change anchor points on multi-series chart |
| 12876 | Fixed | DrawingTool | Resolved an error that occurred when drawing a vertical line to an empty panel |
| 13011 | Fixed | DrawingTool, Templates | Saving a brush to a Text drawing tool's default template overrode brush values passed into NinjaScript |
| 12947 | Fixed | eSignal, Data | For some instruments level II data would not update |
| 12932 | Fixed | FXCM, Orders | Resolved a scenario where a filled order showed as active/working |
| 12890 | Fixed | Indicator | Auto Scale option within the Price Line indicator was showing last price as "0" on a reconnect |
| 12894 | Fixed | Indicator | Volume profile indicator did not draw for non-intraday periods |
| 12903 | Fixed | Indicator | Displacement did not work properly in Regression Channel indicator |
| 12907 | Fixed | Indicator | Indicators with multiple DataSeries could intermittently see 'null' reference during 'SetState' |
| 12925 | Fixed | Indicator | Woodies CCI had incorrect panel values |
| 12980 | Fixed | Indicator | The Darvas indicator was not creating a well formed Darvas box |
| 13004 | Changed | Indicator | Updated CandleStickPattern indicator to be consistent and optimized |
| 12946 | Fixed | Indicator, Chart | When Indicators were active and multiple linked charts switch instruments at the same time an error occurred |
| 12866 | Fixed | Instruments | Splits were not sorted when changes were applied in instruments dialog |
| 12881 | Fixed | Instruments | Pop up dialog for if instrument had server changes had incorrect icon + label text |
| 12915 | Fixed | Instruments | Resolved a scenario where option to add an instrument to a list displayed the wrong instrument if using multiple tabs |
| 12962 | Fixed | Instruments | Resolved a scenario where if an instrument was incorrectly defined it caused a crash rather than just an error |
| 12940 | Fixed | Interactive Brokers | Currently building daily bar was missing for forex instruments |
| 12865 | Fixed | Interactive Brokers, Data | Level II data for live accounts was different than paper accounts and Trader Workstation |
| 12889 | Fixed | Interactive Brokers, Data | Real-time forex bid tick data differed from real-time last tick data |
| 12987 | Fixed | Interactive Brokers, Orders | Resolved an error that could occur if an order was terminal and then was also attempted to be canceled |
| 12848 | Fixed | IQFeed | Optimized loading logic so that historical data which is not available is not requested, resulting in a longer load time |
| 12841 | Fixed | Market Analyzer | Indicator column logic did not support DisplayName override |
| 12845 | Fixed | Market Analyzer | Columns would format to the instrument currency instead of the account currency |
| 12853 | Changed | Market Analyzer | Added up and down action buttons to configured section in the object dialog |
| 12990 | Fixed | Market Analyzer | Total Row was not formatted the same way as column data |
| 12630 | Fixed | NinjaScript | Charts could stop loading and display return to present icon in some scenarios in which a script quickly updates a draw object |
| 12917 | Fixed | NinjaScript | Resolved a null check for areaBrushDevice that could result in an error |
| 12948 | Fixed | NinjaScript | Indicators window could list duplicate instances for strategy added indicators |
| 12964 | Fixed | NinjaScript | Block comments on non-NinjaScript properties caused them to be treated as NinjaScriptProperty in generated code |
| 12966 | Fixed | NinjaScript | If the Dot Dash Style was selected for a NinjaTrader.Gui.Stroke the stroke would not render |
| 12976 | Fixed | NinjaScript | The ReadOnly attribute did not work for strategies on a chart |
| 12280 | Fixed | NinjaScript Editor | Collapsed regions could expand unexpectedly when editing document |
| 12875 | Changed | NinjaScript Editor | Updated Actipro (Third Party Component used for NinjaScript Editor and Output Window) |
| 13013 | Fixed | NinjaScript, Market Analyzer | Load data based on bars did not work for a multi-time frame series indicator |
| 12844 | Fixed | Order Flow + | Depth Map and Trade Detector were incorrectly available in the Market Analyzer and Strategy Builder |
| 12892 | Fixed | Order Flow + | Resolved support for some bar types on Volumetric and made code more consistent with other scripts |
| 12920 | Fixed | Order Flow + | Order Flow Cumulative Delta did not work on non-intraday charts |
| 12944 | Fixed | Order Flow + | Order Flow indicators no longer suspend when inactive as it could affect the values |
| 13008 | Fixed | Order Flow + | Order Flow Cumulative Delta on Line Break charts would not form new bars |
| 13010 | Fixed | Output Window | Resolved an error that occurred when a window was closed that was printing many lines |
| 12656 | Fixed | Performance | Resolved a rare scenario in which quickly working with indicators could increase memory that would not clear |
| 12850 | Fixed | Playback | When no Playback data is available to download for the selected instrument/date, there was no message saying no data |
| 12843 | Fixed | Playback, NinjaScript | Replaying historical data and using Position.GetUnrealizedProfitLoss could cause Playback to lockup |
| 12836 | Fixed | ShareAdapter | Facebook share service discontinued |
| 12919 | Fixed | Simulator | Newly created local simulation accounts now have a prefix of 'Sim' |
| 12918 | Fixed | Strategy | Resolved a scenario where a high fill resolution series that is more granular than primary bars prevented limit orders from filling |
| 12830 | Fixed | Strategy Analyzer | Walk forward optimization displayed chart with incorrect series |
| 12905 | Fixed | Strategy Analyzer | Small values were not populating properly in the results grid when displaying percent data |
| 12927 | Fixed | Strategy Analyzer | Profit values did not display consistently when display type was set to percent |
| 12928 | Fixed | Strategy Analyzer | When running a strategy with slippage set to 0, it will show in the optimizer row but not in the summary for that row |
| 12989 | Fixed | Strategy Analyzer | Performance metric value R2 was not functioning as expected |
| 12901 | Fixed | Strategy Builder | Configurations using input variables in the condition builder could result in an error |
| 12902 | Fixed | Strategy Builder | Setting stop or target values to an indicator resulted in an error |
| 12984 | Fixed | Strategy Builder | When making an indicator created in the Condition Builder plot to a chart, the reference to the plotting indicator was lost in NinjaScript |
| 12997 | Fixed | Strategy Builder | Combined Results were lost when duplicated to new tab |
| 13001 | Fixed | Strategy Builder | Resolved an error that could occur when viewing code that used counter conditions |
| 12887 | Fixed | SuperDOM | SuperDOM Quantity Pad 'Apply' button did not update pop-up list when pressed |
| 12988 | Fixed | SuperDOM | Columns could reduce width when connecting and selecting accounts |
| 13002 | Fixed | SuperDOM, Playback | Volume Column did not reload data in Playback |
| 12885 | Fixed | TD Ameritrade, Historical Data Window | Daily historical data downloaded today's data with tomorrow's timestamps |
| 12991 | Fixed | Trade Performance, Playback | Was unable to filter out Playback execution once disconnected |
| 12922 | Fixed | Window Linking | When using the arrow keys in the Alerts dialog to switch instruments, linked windows would not switch instruments |
| 11176 | Fixed | Workspaces | A crash could occur when switching workspaces while printing to output and connecting |

**8.0.15.1 Release Date**

August 1, 2018

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 13041 | Fixed | Order Flow + | Order Flow Volume Profile reverted custom right side margin when applied |
| 13049 | Fixed | Order Flow + | Restoring Order Flow Volume Profile drawing tool instance from workspace would sometimes fail |
| 13050 | Fixed | Order Flow + | Order Flow Volume Profile Indicator could include more underlying minute data then expected |
| 13051 | Fixed | Order Flow + | Order Flow Volume Profile could throw an exception when multiple load requests were pending |
| 13052 | Fixed | Interactive Brokers, Data | Reverted fix for 12865 due to unexpected side effects |
| **Navigation:**  [Release Notes](https://ninjatrader.com/support/helpGuides/nt8/release_notes.htm) > [8.0](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) >  **8.0.3.0 Release Notes** | | | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/8_0_4_0.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/8_0_2_0.htm) |

**Release Date**

January 9, 2017

|  |
| --- |
| **Attention existing NinjaTrader 8 Users:**As a consequence of bug fix in 8.0.3.0, all DataSeries Trading Hours Templates contained in saved workspace(s) will be reset to factory default settings  “<Use Instrument Settings>”. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 10922 | Fixed | ATM Strategies, NinjaScript | Closing position via AtmStrategyClose method prevented RealizedPnL from updating correctly |
| 10876 | Added | Backup & Restore | Added restart message when restoring |
| 10907 | Fixed | Barchart | Updated DLL version to 1.1.0.9 |
| 10814 | Fixed | Bars | UTC +2 time zone usage created erroneous bars on a chart |
| 10239 | Fixed | Bars | After editing workspace and restarting at times there was an unable to clear cache message |
| 10864 | Fixed | Chart | A crash occurred when creating daily charts in  the evening |
| 10832 | Fixed | Chart | Removing indicator from multi-series chart removed unrelated data series as well |
| 10959 | Fixed | Chart | Center price on scale setting could cause crash on empty charts |
| 10957 | Fixed | Chart | AddOns and Caption Bar alignment was inconsistent |
| 10945 | Fixed | Chart | Changing axis line width caused incorrect margins |
| 10935 | Fixed | Chart | Chart was enlarged when restore from preview to secondary monitor |
| 10916 | Fixed | Chart | Bar spacing was not restoring correctly on multi-series chart |
| 10906 | Fixed | Chart | Indicator displacement did not auto scale correctly |
| 10893 | Fixed | Chart | Chart could scroll to the right unexpectedly |
| 10881 | Fixed | Chart | Chart rendering was failing after rollover of contracts |
| 10877 | Fixed | Chart | Trend Channel parallel line did not snap |
| 10858 | Fixed | Chart | Loading text did not stay in place for secondary series |
| 10815 | Fixed | Chart | Tab name variable tool tips were not defined |
| 10912 | Fixed | Chart Trader | Arrow keys moved chart instead of changing order quantity |
| 10969 | Fixed | Chart Trader | Right click order menu showed invalid order types for some connections |
| 10780 | Fixed | Chart Trader | Resolved a scenario where you could remove a DataSeries which has Scaled Justification Right and Chart Trader orders would no longer display |
| 10925 | Fixed | Chart, Bars | In some scenarios parts of Monday's sessions were not plotting |
| 10851 | Fixed | Chart, DrawingTool | Drawing objects 'attached to' was lost when moving to overlay scale |
| 10720 | Fixed | Chart, Indicator | Editing indicators while chart is still loading at times cause a lock up |
| 10842 | Fixed | Chart, Window Linking | Incorrect chart tab was changing instruments with linked Market Analyzer |
| 10868 | Fixed | Connections, NinjaScript | Disconnect Delay Seconds was not recognizing reconnect |
| 10926 | Fixed | Control Center | At times NinjaTrader could lock up on start up |
| 10857 | Fixed | Control Center | Tab properties font was not saving checked boxes |
| 10900 | Added | Data Grids | Drag column away to delete was implemented on all grids |
| 10812 | Fixed | Database | Resolved a scenario where changing the NinjaTrader language caused a trading hours error |
| 10847 | Fixed | DrawingTool | Draw.Region was shading incorrectly when using displacement |
| 10845 | Fixed | DrawingTool | There was an inconsistent Z-Order between charts for global draw object |
| 10833 | Fixed | DrawingTool | TrendChannel was not properly drawing when attached to all charts by default |
| 10831 | Fixed | DrawingTool | Occasionally there were exception on removing all drawing objects |
| 10984 | Fixed | FX Board | Unhandled exception occurred when disconnecting Kinetick with open FXBoard open |
| 10963 | Fixed | FX Board | Instrument drop-down were not changing color for selected tile |
| 10953 | Fixed | FXCM | XAUUSD and XAGUSD CFD had incorrect symbol mapping |
| 10809 | Fixed | FXCM | In somecases orders could become stuck |
| 10727 | Fixed | FXCM | After a lost connection a crash could occur |
| 10601 | Changed | FXCM | Updated API version to ForexConnect 1.4.1 |
| 8490 | Added | FXCM | Added GTD Order Support |
| 10946 | Fixed | FXCM, Historical Data Window | Control Center locked up when closing Historical Data window while downloading data |
| 10830 | Fixed | Historical Data Window | Historical data manager text was not readable when selected |
| 10885 | Fixed | Hot List Analyzer | @HOTLIST was not present in resources |
| 10834 | Fixed | Indicator | LinReg generated error with period of 1 |
| 10883 | Fixed | Indicator, SuperDOM | Indicator's Input series Price type was not saving |
| 10939 | Fixed | Instruments | Stock list with unsupported characters could be imported but not edited |
| 10936 | Fixed | Instruments | EUREX quarterly rollover dates do not match other EUREX instruments |
| 10928 | Fixed | Instruments | Contract Month had invalid date format |
| 10920 | Fixed | Interactive Brokers | FA accounts could receive position for unknown symbol |
| 10897 | Fixed | Interactive Brokers | Requesting 1 day of historical data downloaded 2 days |
| 10853 | Changed | Interactive Brokers | Symbol mapping updated for ICE TF Point Value change |
| 10824 | Changed | Interactive Brokers | Updated Traders Workstation to 960.2g |
| 10792 | Fixed | Interactive Brokers | Bad order state occurred when order blocked due to TWS precautionary settings |
| 10846 | Fixed | Kinetick | Constant connect/disconnect could occur when PC clock was out of sync |
| 10604 | Fixed | Kinetick | Historical data was unexpectedly throttled |
| 10944 | Fixed | Market Analyzer | Template colors were not restored after closing and opening a workspace |
| 10909 | Fixed | Market Analyzer | Removed invalid conditions when using Alerts |
| 10849 | Fixed | Market Analyzer | Suspended indicators did not catch up until bar closes after restored |
| 10554 | Changed | Market Analyzer | Added log error when using indicators not compatible with end of day data |
| 10335 | Fixed | Market Analyzer | Typing label text which exceeded window length triggered instrument search box |
| 10918 | Fixed | NinjaScript | LockRecursionException occurred on reloading NinjaScript after changing added series |
| 10929 | Fixed | NinjaScript | IndexOutOfRange exception occurred when there was an added series in hosted indicator |
| 10902 | Fixed | NinjaScript | AddDataSeries with specified template was not working as expecting in all scenarios |
| 10870 | Fixed | NinjaScript | Creating a new strategy while in a bad compile state halted NinjaTrader |
| 10861 | Fixed | NinjaScript | Removing an indicator then reloading a chart resulted in an exception |
| 10841 | Fixed | NinjaScript | High order fill resolution did not load expected bars when 'bars back' is used |
| 10840 | Changed | NinjaScript | Methods .PlaySound/.SendMail/.Share now can be triggered in State==.Active |
| 10758 | Fixed | NinjaScript | Exceptions could be generated by Finalized NinjaScript |
| 10839 | Fixed | NinjaScript, Strategy Analyzer | MAE was incorrect for multi-series strategies |
| 10884 | Fixed | Other | First In Product Announcement was modal only to Control center, other ones are modal across the app as expected |
| 10948 | Fixed | Playback | Controller's time-stamp lagged behind bars when paused |
| 10911 | Fixed | Playback | Slider became stuck if only two days were downloaded |
| 10888 | Fixed | Playback | At times Go To could not select time |
| 10866 | Fixed | Playback | Exception was occurring when disconnecting if End Date calendar was open |
| 10291 | Fixed | Playback | Migration from NinjaTrader 7 playback data had incorrect time offset |
| 10938 | Fixed | Property Grids | Orders Grid Properties allowed removing of Instrument column |
| 10838 | Fixed | ShareAdapter | Stocktwits share service log-in was throwing errors and was unresponsive |
| 10982 | Fixed | Strategy Analyzer | Assigned account could unexpectedly change on subsequent runs in certain scenarios. |
| 10952 | Fixed | Strategy Analyzer | Move to new window was available while running which resulted in an error |
| 10894 | Fixed | Strategy Analyzer | Null reference could occur when switching between backtest and optimization |
| 10886 | Fixed | Strategy Analyzer | Strategy was running two instances on single run |
| 10850 | Fixed | Strategy Analyzer | Repeatedly running a backtest caused memory to increase until there was a crash |
| 10843 | Fixed | Strategy Analyzer | There were incorrect tab names |
| 10826 | Fixed | Strategy Analyzer | Sorting pinned logs caused pins to be lost |
| 10967 | Fixed | Strategy Builder | IsFalling and IsRising was calling the bar index |
| 10949 | Fixed | Strategy Builder | MACD's plot name was Default rather than Macd |
| 10932 | Fixed | Strategy Builder | Unhandled Exception occurred when opening Strategy Building with compile errors |
| 10836 | Fixed | Strategy Builder | Removed invalid settings |
| 10816 | Fixed | Strategy Builder | An unhandled exception could occur when adding Actions in a certain scenario |
| 10913 | Fixed | SuperDOM | Dynamic SuperDOM was not always shown as a menu item |
| 10821 | Fixed | SuperDOM | Removed invalid order references for APQ column |
| 10921 | Fixed | TD AMERITRADE | External orders at times caused exceptions |
| 10672 | Fixed | TD AMERITRADE | Linked account hanged on connection attempt |
| 10954 | Fixed | Tool Tips | Getting started arrows were hard to see |
| 10903 | Fixed | Trade Performance | Journal entries were not hidden/shown based on generated date range |
| 10891 | Fixed | Trade Performance | Cut, copy, and paste was not working in all areas |
| 10890 | Fixed | Trade Performance | Journal entries were not properly logging notes |
| 10962 | Fixed | UI | Help > Email Support not saving email address |
| 10914 | Fixed | UI | End keyboard button switched chart tabs after "F" (fixed) button is pressed |
| 10844 | Fixed | UI | Resolved scenarios where in product announcement would block migration |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) > [Educational Resources](https://ninjatrader.com/support/helpGuides/nt8/educational_resources.htm) > [Developing Indicators](https://ninjatrader.com/support/helpGuides/nt8/developing_indicators.htm) > [Beginner - Using price variables](https://ninjatrader.com/support/helpGuides/nt8/beginner_-_using_price_variabl.htm) >  **Entering Calculation Logic** | | | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/set_up4.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/beginner_-_using_price_variabl.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/compiling.htm) |

The [OnBarUpdate()](https://ninjatrader.com/support/helpGuides/nt8/onbarupdate.htm) method is called for each incoming tick, or on the close of a bar (if enabled) when performing real-time calculations, and is called on each bar of a [Bars](https://ninjatrader.com/support/helpGuides/nt8/bars.htm) object when re-calculating the indicator (For example, an indicator would be re-calculated when adding it to an existing chart that has existing price data displayed). This is the main method used for indicator calculations, and we will calculate our core indicator logic (testing to see if a Close price on a specified bar was greater than the previous Close price) within this method.

**Adding the Condition and Assigning the Plot Value**

Enter the following code in the OnBarUpdate() method in the NinjaScript Editor:

| ns |
| --- |
| Values[0][BarsAgo] = (Close[BarsAgo] > Close[(BarsAgo + 1)]) ? (High[BarsAgo] + (5 \* TickSize)) : (Low[BarsAgo] - (5 \* TickSize)); |

Although the code above fits on a single line, it is doing several things. Firstly, it is important to understand the structure that we are using in this statement. We are using a [Ternary Operator](https://msdn.microsoft.com/en-us/library/ty67wk28.aspx), which provides a way to assign one of two values to a variable based on a condition. We begin by stating that we wish to assign a value to the indicator plot at a bar index corresponding to BarsAgo. We do this by using [Values](https://ninjatrader.com/support/helpGuides/nt8/values.htm), which is a collection holding values for all plots configured in the indicator:

| ns |
| --- |
| Values[0][BarsAgo] = |

Next, we add a condition to test. In this case, we are testing to see whether [Close](https://ninjatrader.com/support/helpGuides/nt8/close.htm) at a bar index corresponding to the value of BarsAgo was greater than Close at a value of BarsAgo + 1. If BarsAgo was set to 5, for example, this would compare Close[5] to Close[6]:

| ns |
| --- |
| Values[0][BarsAgo] = (Close[BarsAgo] > Close[(BarsAgo + 1)]) ? |

If the condition evaluates to true, then the first expression will be run (the expression on the left side of the colon ":"), which will assign the value of the indicator plot to the [High](https://ninjatrader.com/support/helpGuides/nt8/high.htm) price of the specified bar, plus five ticks. We obtain the tick size value for the configured instrument via the [TickSize](https://ninjatrader.com/support/helpGuides/nt8/ticksize.htm) property:

| ns |
| --- |
| Values[0][BarsAgo] = (Close[BarsAgo] > Close[(BarsAgo + 1)]) ? (High[BarsAgo] + (5 \* TickSize)) : |

if the condition evaluates to false, then the second expression will be run (the expression on the right side of the colon ":", which will assign the value of the indicator plot to the [Low](https://ninjatrader.com/support/helpGuides/nt8/low.htm) price of the specified bar, less five ticks:

| ns |
| --- |
| Values[0][BarsAgo] = (Close[BarsAgo] > Close[(BarsAgo + 1)]) ? (High[BarsAgo] + (5 \* TickSize)) : (Low[BarsAgo] - (5 \* TickSize)); |

The core indicator logic is now in place, but running this code as it is can result in an "Index out of range" exception. Since we are looking a certain number of bars back in time, we need to make sure that there are always enough bars on the chart for us to look back. For example, if BarsAgo were set to 5, then we would be comparing the value of five bars ago to the value of six bars ago, but on Bars # 1, 2, 3, 4, or 5, at which point we do not have five or six bars to look back, the indicator will cause an error. To resolve this, we will add a condition which will prevent the core calculations from running unless we know there are enough bars on the chart. Add the following line just above the line you have been working on throughout this page:

| ns |
| --- |
| if(CurrentBar < BarsAgo + 1)   return; |

This line says, "if there is not a number of bars equal to one number greater than the value of BarsAgo, then exit OnBarUpdate()."

Now that everything is in place, your class code should look as below. You are now ready to [compile the indicator](https://ninjatrader.com/support/helpGuides/nt8/compiling.htm) and configure it on a chart.

| ns | |
| --- | --- |
| public class PriceVariableTutorial : Indicator {   protected override void OnStateChange()   {       if (State == State.SetDefaults)       {           Description                 = @"NinjaScript Price Variables Tutorial";           Name                       = "PriceVariableTutorial";           Calculate                   = Calculate.OnBarClose;           IsOverlay                   = false;           DisplayInDataBox           = true;           DrawOnPricePanel           = true;           DrawHorizontalGridLines     = true;           DrawVerticalGridLines       = true;           PaintPriceMarkers           = true;           ScaleJustification         = NinjaTrader.Gui.Chart.ScaleJustification.Right;           //Disable this property if your indicator requires custom values that cumulate with each new market data event.           //See Help Guide for additional information.           IsSuspendedWhileInactive   = true;           BarsAgo                 = 0;           AddPlot(Brushes.Orange, "MyPlot");       }       else if (State == State.Configure)       {       }   }     protected override void OnBarUpdate()   {       if(CurrentBar < BarsAgo + 1)           return;               Values[0][BarsAgo] = (Close[BarsAgo] > Close[(BarsAgo + 1)]) ? (High[BarsAgo] + (5 \* TickSize)) : (Low[BarsAgo] - (5 \* TickSize));   }    #region Properties   [Range(0, int.MaxValue)]   [NinjaScriptProperty]   [Display(Name="BarsAgo", Description="How many bars ago to use for the plot value", Order=1)]   public int BarsAgo   { get; set; }     [Browsable(false)]   [XmlIgnore]   public Series<double> MyPlot   {       get { return Values[0]; }   }   #endregion   } | |
| **Navigation:**  [Operations](https://ninjatrader.com/support/helpGuides/nt8/operations.htm) > [Charts](https://ninjatrader.com/support/helpGuides/nt8/charts.htm) >  **Working with Price Data** | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/chart_objects.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/charts.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/working_with_multiple_data_series.htm) |

A **Data Series** represents a series of price data, which can be displayed on a chart using one of several [Bar Types](https://ninjatrader.com/support/helpGuides/nt8/bar_types.htm) and **Chart Styles**. One or more **Data Series** will be applied to a new chart when it is created, and additional **Data Series** can be added, edited, or removed via the **Data Series** window.

|  |
| --- |
| playVideo |

tog_minus        [Understanding the Data Series Window](javascript:HMToggle('toggle','UnderstandingTheDataSeriesWindow','UnderstandingTheDataSeriesWindow_ICON'))

|  |  |
| --- | --- |
| The **Data Series** window is used to configure the **Data Series** within a chart, edit **Data Series** parameters, and save default values for different **Period Types**.    **Accessing the Data Series Window**  There are multiple ways to access the **Data Series** window:    •Select the **New** menu from the NinjaTrader **Control Center**, then select the **Chart** menu item.  •Right mouse click in the chart background and select the **Data Series** menu item.  •Use the default CTRL+F [Hot Key](https://ninjatrader.com/support/helpGuides/nt8/hot_key_manager.htm) from an open chart.  •Double left mouse click on a **Data Series** within the chart.  •Right mouse click on a selected **Data Series** within a chart, then select the **Properties** menu item.    **Sections of the Data Series Window**  The image below displays the four sections of the **Data Series** window.    PriceData1    **1.Instrument Selector**  **2.Data Series** currently applied to the chart  **3.**Selected **Data Series'** parameters  **4.**Saved Chart Templates that can be applied to the new chart. See the [Saving Chart Defaults and Templates](https://ninjatrader.com/support/helpGuides/nt8/saving_chart_defaults_and_templates.htm) page for more information.     |  | | --- | | **Note**:  If a [Chart Template](https://ninjatrader.com/support/helpGuides/nt8/saving_chart_defaults_and_templates.htm) is selected, settings from that template will take precedence over any settings manually configured on the **Data Series**.  For example, **Trading Hours** currently configured will be ignored, and the chart will use the **Trading Hours** which were saved in the**Chart Template.** | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_price_data.htm#UnderstandingTheDataSeriesWindow)

tog_minus        [How to add a Data Series](javascript:HMToggle('toggle','HowToAddADataSeries','HowToAddADataSeries_ICON'))

|  |  |
| --- | --- |
| **Adding a Data Series**  Multiple **Data Series** objects can be applied within a single chart. A new panel is automatically created for each **Data Series** added, unless the "Panel" property is manually changed to an existing panel. There are multiple ways to add a **Data Series** to a chart using the **Data Series** window:    1.Use the **Instrument Selector** dropdown menu to select a recently used or pinned instrument, or any instrument in an **Instrument List**.  2.Type the instrument symbol (including the contract month for futures instruments) directly into the **Instrument Selector**, then press the "Enter" key.  3.Left mouse click on the magnifying glass icon next to the **Instrument Selector**. In the window that appears, use the search field to search available instruments by symbol or description, then double left mouse click on an instrument in the search results to add it to the list of applied **Data Series**.    The added **Data Series** will now be visible in the list in the "Applied" section, allowing you to change any parameters to desired values (see the "*How to edit Data Series parameters*" section below).     |  | | --- | | **Tip**: A **Data Series** can also be added by typing directly into an open chart. Type the plus symbol (+) followed by the instrument symbol, contract month for Futures, and appropriate interval value. For example, typing "+ES ##-## 5M" will add a 5 minute ES continuous contract **Data Series** to the selected chart (See the "*How to change a Data Series*" section below for more information). |       PriceData2    In the image above, we can use the [Instrument Selector](https://ninjatrader.com/support/helpGuides/nt8/usingtheinstrumentselector.htm) to add a recently viewed or pinned instrument, as well as any instruments in an **Instrument List**. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_price_data.htm#HowToAddADataSeries)

tog_minus        [How to edit Data Series parameters](javascript:HMToggle('toggle','HowToEditDataSeriesParameters','HowToEditDataSeriesParameters_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Editing a Data Series**  A **Data Series** object's parameters are available to configure within the **Data Series** window once it has been added to a chart (see the "*How to add a Data Series*" section above).    To edit **Data Series** parameters:    1.Open the **Data Series** window (see the "*Understanding the Data Series window*" section above).  2.Select the **Data Series** you would like to edit in the "Applied" section.  3.Once selected, the **Data Series** parameters will be available to edit on the right hand side.    PriceData3    Available **Data Series** parameters can be found in the list below:    **Data Series Parameters**   |  |  | | --- | --- | | Price based on | Sets the type of market data used to drive the **Data Series** | | Type | Sets the bar type of the **Data Series**. See the [Bar Types](https://ninjatrader.com/support/helpGuides/nt8/bar_types.htm) page for more information. | | Value | Sets the **Data Series** value, based on the selected **Bar Type** | | Tick Replay | Enables Tick Replay on the selected **Data Series**. This option will only display when "Show Tick Replay" is enabled in the [Options](https://ninjatrader.com/support/helpGuides/nt8/options.htm) window | | Load data based on | Determines how much data is loaded based on number of bars, number of days, or a custom date range. | | Days Back / Bars Back / Start Date | Sets the value for the amount of historical data to load, based on the "Load Data Based On" setting. The label on this property will change based upon what you have selected for the "Load Data Based On" property. | | End date | Sets the end date of the chart. If the specified end date is within the range of an applied **Trading Hours** template whose end time falls on a future date, then the Chart will end on that future date. | | Trading Hours | Sets the **Trading Hours** template to be used for the **Data Series**. See the [Trading Hours](https://ninjatrader.com/support/helpGuides/nt8/trading_hours.htm) page for more information. | | Break at EOD | When enabled bars will be cut off at the end of the **Trading Hours** session regardless of whether it is fully complete. So a 10 range bar may close with a range of 3 or a 4 hour bar may close after 2 hours. When disabled, such a bar will continue to develop until it is complete, potentially causing it to post outside of the **Trading Hours** session. For more information, see the [Break at EOD](https://ninjatrader.com/support/helpGuides/nt8/break_at_eod.htm) page. | | Chart Style | Sets the style of the bars. Custom [Chart Styles](https://ninjatrader.com/support/helpGuides/nt8/chart_style.htm) can be created via NinjaScript to extend the pre-built list. | | Bar Width | Sets the width of the bars drawn on the chart | | Additional Chart Style Options | Additional options for configuring bar colors and related properties will be displayed beneath the Bar Width property, depending on which **Chart Style** you have selected. | | Auto Scale | When enabled, the **Data Series** will be part of the chart's auto scaling.    •In case the chart is set to a fixed scale, this property has no effect.  •In case there are no objects on the chart which have this property set to true, the first chart object will be used for the chart's auto-scaling. | | Center on Price Scale | When enabled, the current price will be centered on the price axis, and all visible historical bars will be scaled accordingly | | Display in Data Box | Enables or Disables the display of the selected **Data Series** in the [Data Box](https://ninjatrader.com/support/helpGuides/nt8/data_box.htm) | | Label | Sets the label text to be displayed in a chart panel when more than one **Data Series** has been applied to a chart. This can be left blank to remove the label entirely. | | Panel | Sets the panel in which the selected **Data Series** will be plotted. When more than one **Data Series** has been added to a chart, all but the first **Data Series** in the list will provide the option to plot in a **New Panel** in the "Panel" field. | | Price Marker | Expanding this property will allow you to change the color for the price markers on the chart, as well as enable or disable the price markers' visibility. | | Scale justification | Sets the scale on which the **Data Series** will be plotted. Possible values are "Right," "Left," and "Overlay" | | Show Global Draw Objects | Sets whether **Global Drawing Objects** will be displayed for this **Data Series**. See the "*Understanding local vs. global drawing objects*" section of the [Working with Drawing Tools & Objects](https://ninjatrader.com/support/helpGuides/nt8/working_with_drawing_tools__ob.htm) page for more information. | | Trading Hours Break Line | Sets the color, dash style, and width of the **Trading Hours** break line plotted on the chart for the selected **Data Series** | | Color for Executions - Buy | Sets the color for Buy-side execution markers | | Color for Executions - Sell | Sets the color for Sell-side execution markers | | NinjaScript Strategy Profitable Trade Line | Sets the color, dash style, and width for the lines connecting entries and exits of profitable trades taken by a NinjaScript strategy | | NinjaScript Strategy Unprofitable Trade Line | Sets the color, dash style, and width for the lines connecting entries and exits of unprofitable trades taken by a NinjaScript strategy | | Plot Executions | Sets the plotting style of the trade executions.     |  | | --- | | **Note**: Real-time executions are timestamped based on the timezone set in the "General" section of the **Options** window, which can be accessed from the [Tools](https://ninjatrader.com/support/helpGuides/nt8/tools_menu.htm) menu in the **Control Center**. Please see the [How Trade Executions are Plotted](https://ninjatrader.com/support/helpGuides/nt8/how_trade_executions_are_plott.htm) page for more information. | | | Execution marker size | Sets the size of execution markers |     **Saving Data Series Parameters as Default**  You can optionally save your customized **Data Series** parameters as default. Defaults are saved based on the **Interval Type** selected. Saving defaults will recall your customized settings the next time you add a **Data Series** with that specific **Interval Type** to a chart. Please see the [Saving Chart Defaults and Templates](https://ninjatrader.com/support/helpGuides/nt8/saving_chart_defaults_and_templates.htm) page for more information. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_price_data.htm#HowToEditDataSeriesParameters)

tog_minus        [How to change a Data Series](javascript:HMToggle('toggle','HowToChangeADataSeries','HowToChangeADataSeries_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Series** can be edited in several ways after being added to a chart.    **Changing an Instrument via the Chart Toolbar**  To change an instrument using the chart toolbar:    1.Left mouse click on the instrument drop down menu in the chart toolbar  2.Select a recent or pinned instrument from the top of the list, or expand any of the **Instrument Lists** for additional selections (for more information about editing **Instrument Lists**, see the [Instrument Lists](https://ninjatrader.com/support/helpGuides/nt8/instrument_lists.htm) page).    **Using the Interval Selector**  The **Interval Selector** can be used to change a **Data Series** interval directly from the chart toolbar. The **Interval Selector** comes pre-populated with commonly used intervals, but you can add additional intervals of your choice at any time. To access the**Interval Selector**, left mouse click the dropdown menu displaying the currently selected interval, located next to the instrument dropdown menu on the chart toolbar. To change the currently selected interval, select any of the values corresponding to the row labeled with your desired interval type. For example, to switch to a 5,000 **Volume** interval, click the "5000" option in the "Volume" row.    PriceData4    **Adding Intervals to the Interval Selector**  To add a new interval to the **Interval Selector**, first click the **Configure** option. The **Configure** window that appears is separated into two sections. In the "Intervals" section on the left side, you can select any existing **Interval Type** to view, add, edit, or remove any specific interval value set up for that interval type. In this section, you can add new **Interval Types** to the list via the **add** option, remove an **Interval Type** from the list via the **remove** option, or move **Interval Types** higher or lower in the list via the **up** and **down** options. In addition to the **Interval Types** already available, you can add Heiken Ashi, Kagi, Line Break, Point and Figure, or Renko to the list.    With an **Interval Type** selected in the "Intervals" section, you can manage the specific intervals available for that type in the "Values" section. To add a specific interval to the list for a specific **Interval Type**, select the **add** option. A window will appear, in which you can set the label and **Data Series** options to be used when that interval is selected:    PriceData5    1. The "Label" field sets the label that will be displayed in the **Interval Selector** for this interval. Entering "@VALUE" in this field will display the value entered in the "Value" field in the section below. Alternatively, you can enter any text or numbers in this field to label the interval.    2. The "Price Based On" field determines whether the underlying **Data Series** will be based upon the Ask, Bid, or Last price for the selected instrument.    3. The "Value" field sets the value to be used for the interval, based on the **Interval Type**.    **Editing, Sorting, and Removing Intervals**  To remove an interval from the list for a specific interval type, first select the interval, then select the **remove** option.    To edit the parameters of an existing interval, select the **edit** option instead.    To change the placement of an interval in the list, first select the interval you wish to move, then select the **up** or **down** options to move it higher or lower in the list. Moving an interval higher in the list will cause it to be displayed further to the left in the **Interval Selector**, and moving it lower in the list will cause it to be displayed further to the right.    PriceData6    The **Configure** window pictured above allows the addition, removal, or editing of interval types and specific intervals in the **Interval Selector**.    **Changing and Adding Instruments and Intervals with the Keyboard**  You can change instruments or intervals by pressing a letter or number key in a selected chart. When a letter or number key is pressed, the **Instrument Overlay** appears. Within the **Instrument Overlay**, you can change the instrument, interval, or chart type by using the formats in the table below and pressing the "Enter" key when finished. If multiple instruments are displayed in the chart, you can change a specific instrument by left mouse clicking to select it before typing. If no instrument is selected, the primary instrument is changed.    PriceData7pri     |  |  | | --- | --- | | To change an instrument: | Type the instrument symbol (Add the contract month for futures instruments). Examples: "ES ##-##" for E-mini S&P 500, "AAPL" for Apple stock, or "EURUSD" for Euro/USD forex pair. | | To change an interval: | Type interval value plus the interval suffix (Value +suffix). Examples: "5M" for 5 minute bars, "100T" for 100 tick bars, "1D" for 1 Day bars,10 etc. | | Available suffixes: | Suffix interval: | | M | Minute | | T | Tick | | V | Volume | | R | Range | | S | Second | | D | Day | | W | Week | | MO | Month | | Y | Year | | RE | Renko | | To change instrument and interval | Type the symbol and interval together. For example, typing "AAPL 5M" will change to a 5 minute chart of Apple stock. | | To add additional series of primary instrument | Type a plus sign (+) plus the interval. For example, typing "+5M" will add a 5 minute **Data Series** of the primary instrument. | | To add additional series of any instrument | Type a plus sign (+) plus the instrument and interval. For example, typing "+AAPL 5M" will add a 5 minute series of Apple stock. If no interval is provided, then the same interval as the primary series will be added. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_price_data.htm#HowToChangeADataSeries)

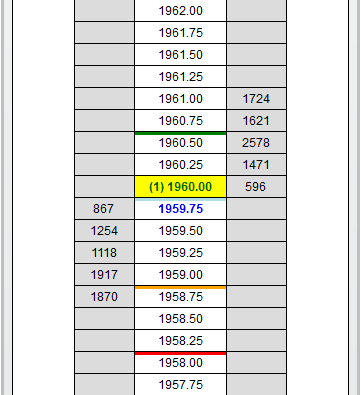
tog_minus        [Removing a Data Series](javascript:HMToggle('toggle','RemovingADataSeries','RemovingADataSeries_ICON'))

|  |
| --- |
| **Removing a Data Series**  There are three ways to remove a **Data Series** from your NinjaTrader chart:    •Open the **Data Series** window *(see the "Understanding the Data Series window" section above). S*elect a **Data Series** from the "Applied" section, then select the **Remove** option, then press the **OK** button to close the **Data Series** window.  •Left mouse click a **Data Series** on your chart to select it, then press the "Delete" button on your keyboard.  •Left mouse click  a **Data Series** on your chart to select it, then right mouse click the **Data Series** and select the **Remove** menu item.    If only one **Data Series** is applied to a chart, it cannot be removed. However, the original **Data Series** added to a chart can be removed *if* there is at least one other **Data Series** is still applied. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_price_data.htm#RemovingADataSeries)

|  |  |
| --- | --- |
| **Navigation:**  [Operations](https://ninjatrader.com/support/helpGuides/nt8/operations.htm) > [Order Entry](https://ninjatrader.com/support/helpGuides/nt8/order_entry.htm) > [SuperDOM](https://ninjatrader.com/support/helpGuides/nt8/superdom.htm) >  **Working with Indicators** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/superdom_templates.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/superdom.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/properties_superdom.htm) |
| playVideo |

The **SuperDOM's** [Price Ladder display](https://ninjatrader.com/support/helpGuides/nt8/price_ladder_display.htm) has the ability to add any number of price action indicators which can be used to visualize and analyze indicator values in relation to the **SuperDOM** display, as well as attaching working orders to the indicator price level for a hand-free trade management system.



NinjaTrader comes with over 30 pre-built indicators which can be added the **SuperDOM**. Indicators can be added, removed and edited via the Indicators window.

tog_minus        [Understanding the Indicators window](javascript:HMToggle('toggle','UnderstandingTheIndicatorsWindow','UnderstandingTheIndicatorsWindow_ICON'))

|  |
| --- |
| The Indicators window is used to add, remove and edit all indicators within a **SuperDOM**.  **Accessing the Indicators Window**  •Right mouse click in the **SuperDOM** select the menu **Indicators**  **Sections of the Indicators Window**  The image below displays the three sections of the Indicators window.    1.List of **Available** indicators (a description of the selected indicator can be viewed by clicking on the info_icon symbol, see the green arrow in the image below)  2.Current indicators **Configured** on the **SuperDOM**  3.Selected indicator's **Properties**    SuperDOM_36 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators_superdom.htm#UnderstandingTheIndicatorsWindow)

tog_minus        [How to add an indicator](javascript:HMToggle('toggle','HowToAddAnIndicator','HowToAddAnIndicator_ICON'))

|  |
| --- |
| **Adding an Indicator**  To add an indicator to a **SuperDOM**:  1.Open the Indicators window *(see the "Understanding the Indicators window" section above)*  2.Left mouse click on the **Available** indicator you want to add and press the **Add** button or simply double click on it  3.The indicator will now be visible in the list of **Configured** indicators  4.The indicator's parameters will now be editable on the right side of the Indicators window *(see the "How to edit an indicator" section below)*    SuperDOM_37 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators_superdom.htm#HowToAddAnIndicator)

tog_minus        [How to edit an indicator's](javascript:HMToggle('toggle','HowToEditAnIndicators','HowToEditAnIndicators_ICON')) parameters

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Editing an Indicator**  You can customize any indicator from the Indicators window:  1.Open the Indicators window (see the*"Understanding the Indicators window"*section above)  2.Highlight the indicator you would like to edit from the list of applied indicators (as shown in the image below).  3.Once highlighted this indicator's parameters will be available to edit on the right hand side.  SuperDOM_38  **Indicator Parameters**  The following parameters are common to all indicators:     |  |  | | --- | --- | | **Data Series** |  | | Input Series | Please see the Input series section for further information. | | Price based on | Sets the type of market data used to drive the Data Series. (Last, Ask, Bid) | | Type | Sets the bar type of the Data Series. (See the [Bar Types](https://ninjatrader.com/support/helpGuides/nt8/bars_type.htm) section of the Help Guide for more information) | | Value | Sets the Data Series value. | | **Time frame** |  | | Load data based on | Determines how much data is loaded based on number of bars, number of days, or a custom date range. | | Bars to load | Sets the number of bars or days to load data. | | End date | Sets the end date of the data used in indicator's calculation | | Trading hours | Sets the Trading hours used for the Data Series. (See the [Trading Hours](https://ninjatrader.com/support/helpGuides/nt8/sessioniterator.htm) section of the Help Guide for more information) | | Break at EOD | Enables or disables the bars being reset at EOD (End Of Day). (See the "[Understanding Historical Data](https://ninjatrader.com/support/helpGuides/nt8/data_by_provider.htm)" section of the Help Guide for more information) | | **Set up** |  | | Calculate | Sets the frequency that the indicator calculates. **On bar close** will slow down the calculation until the close of a bar; **On price change** will calculate on when there has been a change in price; **On each tick** calculate the indicator's value which each incoming tick. | | Maximum bars look back | Max number of bars used for calculating an indicator's value.  The TwoHundredFiftySix setting is the most memory friendly. | | **Visual** |  | | Visible | Sets if the indicator plot is visualized on the display | | **Plots**(...) | Allows you to customize the appearance of the indicator by changing the Color or Thickness |     **Saving an Indicator's Parameters as Default**  You can optionally save your customized indicator's parameters as a default preset. Doing so will recall your customized settings the next time you add this specific indicator to a **SuperDOM**.    Once you have your indictor's properties set to your preference, you can left mouse click on the "**preset**" text located in the bottom right of the properties dialog. Selecting the option "**save**" will save these settings as the default settings used every time you open a new window/tab.    If you change your settings and later wish to go back to the original settings, you can left mouse click on the "**preset**" text and select the option to "**restore**" to return to the original settings. |
| **Indicator Input Series**  The indicator Input Series window allows you to select the input series for your indicator's calculations.  This allows you to configure different data types, such as the High, or Open price, or even calculate your indicators based off of multiple nested indicators.    To access this window, move your mouse over the **Input Series**field, which will change to an "**Edit input...**" button.    1.  You can then select the Close, High, Low, Median, Open, Typical, or Weighted value of any Data Series within a **SuperDOM**.  2.  Additionally, you can also choose another indicator as the input series.  When you select another indicator as the input series, you can define the properties used in the input series for the second indicator. Once you have selected the input series of your choice left mouse click the OK button to exit the Input Series window.    SuperDOM_39 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators_superdom.htm#HowToEditAnIndicators)

tog_minus        [Understanding how indicators are displayed](javascript:HMToggle('toggle','UnderstandingHowIndicatorsAreDisplayed','UnderstandingHowIndicatorsAreDisplayed_ICON'))

|  |  |
| --- | --- |
| **Indicator Display** Once an indicator has been configured and applied to the **SuperDOM**, the indicator plot will be displayed in the **Price** **column** above the corresponding price row.    SuperDOM_45    In the image above, you can see an orange highlighted price row at 1959.75, rounded to nearest price from the calculated EMA indicator (1959.63).  Hovering your mouse cursor over the indicator plot will display a tool tip which will give you details pertaining to input settings of the indicator.     |  | | --- | | **Note:**It is possible for indicators to be calculated out of range of the current **Price Ladder** **Display**.  You can right click on the **SuperDOM** and uncheck **Auto Center** which will allow you to scroll up or down on the **Price Ladder Display** to locate the indicator that has been added. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators_superdom.htm#UnderstandingHowIndicatorsAreDisplayed)

tog_minus        [How to remove an indicator](javascript:HMToggle('toggle','HowToRemoveAnIndicator','HowToRemoveAnIndicator_ICON'))

|  |
| --- |
| **Removing an Indicator**  To remove an indicator from your NinjaTrader **SuperDOM**:    •Open the Indicators window *(see the "Understanding the Indicators window" section above)*, select an indicator from the **Configured** indicators list, press the Remove button, and then press the OK button to exit the Indicators window. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators_superdom.htm#HowToRemoveAnIndicator)

tog_minus        [Custom indicator development](javascript:HMToggle('toggle','CustomIndicatorDevelopment','CustomIndicatorDevelopment_ICON'))

|  |  |
| --- | --- |
| In addition to the over 30 price action indicators that come pre-built with the NinjaTrader application, you also have the ability to create custom indicators of your own.  For example, you could create your own custom multi-series indicators to apply to your NinjaTrader **SuperDOMs**.     |  | | --- | | **Note**:  In order for a custom indicator to show up in the list of available **SuperDOM Indicators**, you must set the [IsOverlay](https://ninjatrader.com/support/helpGuides/nt8/isoverlay.htm) property to**true**in the indicator's**State.SetDefaults.** |       For more information on using NinjaScript to build custom indicators please see the [NinjaScript section](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) of the user help guide. Click [here](https://ninjatrader.com/support/helpGuides/nt8/indicator.htm) to view NinjaScript tutorials.    The option to hire a [NinjaScript Consultant](https://ninjatraderecosystem.com/search-results/?fwp_category=programming-services) to build your custom indicators is also available. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_indicators_superdom.htm#CustomIndicatorDevelopment)

|  |  |
| --- | --- |
| **Navigation:**  [Operations](https://ninjatrader.com/support/helpGuides/nt8/operations.htm) > [Charts](https://ninjatrader.com/support/helpGuides/nt8/charts.htm) >  **Bar Types** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/working_with_multiple_data_series.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/charts.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/chart_styles.htm) |

NinjaTrader supports a large variety of chart **Bar** **Types**. This page explains how each **Bar Type** is created in a chart. Please see the [Working with Price Data](https://ninjatrader.com/support/helpGuides/nt8/working_with_price_data.htm) page for information on how to change **Bar** **Types**.

|  |
| --- |
| **Notes:**  •For some **Bar Types**, the last bar of a session may be built as an incomplete bar due to the session ending before the bar could be completed. Each new session will have bars freshly built beginning from the first tick of the session. For example, the last bar of a session in a 10,000 Volume chart may contain a volume less than 10,000, while the next bar which builds on a new session would contain 10,000 volume. This behavior can be changed via the "Break at EOD" **Data Series** property. For more information, see the [Break at EOD](https://ninjatrader.com/support/helpGuides/nt8/break_at_eod.htm) page.  •When backtesting different **Bar** **Types** (most notably Point and Figure and Renko) the backtest can yield different results than what you would experience in real-time, due to the nature of how the bars are constructed and the possibility of not having enough granular information to simulate what would have happened in real-time. Please see the [Discrepancies: Real-time vs Backtest](https://ninjatrader.com/support/helpGuides/nt8/discrepancies_real-time_vs_bac.htm) page for more information.  •When working with [TickReplay](https://ninjatrader.com/support/helpGuides/nt8/tick_replay.htm), the bars will be built from tick data available through the provider or local repository. For developing NinjaScript objects taking advantage of this option, please see [this link](https://ninjatrader.com/support/helpGuides/nt8/developing_for__tick_replay.htm). |

tog_minus        [Understanding Tick bars](javascript:HMToggle('toggle','UnderstandingTickBars','UnderstandingTickBars_ICON'))

|  |
| --- |
| **Tick Bars**  A **Tick** bar is based on a specific number of ticks. A bar will continue to develop until the specified number of ticks is reached. The next tick will then result in a new bar being created.    BarTypes1    1. Each historical bar in the the 500 Tick chart shown above plots a total of 500 ticks.  2. The "Tick Counter" [indicator](https://ninjatrader.com/support/helpGuides/nt8/working_with_indicators.htm) has been applied to the chart to show the number of ticks remaining in the current bar. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?bar_types.htm#UnderstandingTickBars)

tog_minus        [Understanding Volume bars](javascript:HMToggle('toggle','UnderstandingVolumeBars','UnderstandingVolumeBars_ICON'))

|  |
| --- |
| **Volume Bars**  A **Volume** bar is based on a specific number of units traded (volume). A bar will continue to develop until the specified volume is reached, and once that level is surpassed, a new bar will be created.    BarTypes2    1. Each historical bar in the 10,000 **Volume** chart shown above contains a volume of 10,000 contracts.  2. This is verified by the "VOL" [indicator](https://ninjatrader.com/support/helpGuides/nt8/working_with_indicators.htm) plotted below the price bars and the Volume displayed in the [Mini Data Box](https://ninjatrader.com/support/helpGuides/nt8/data_box.htm). |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?bar_types.htm#UnderstandingVolumeBars)

tog_minus        [Understanding Range bars](javascript:HMToggle('toggle','UnderstandingRangeBars','UnderstandingRangeBars_ICON'))

|  |  |
| --- | --- |
| **Range Bars**  A **Range** bar is based on a specified tick price range. The bar will continue to develop until the price range is broken, at which point a new bar will be created.     |  | | --- | | **Note**: A tick in this instance is different from a tick in a **Tick** bar described in the sub-section above. A tick in a **Tick** bar represents the point at which an actual trade occurred, whereas a tick in a **Range** bar represents a price increment, or a movement on the price axis of the chart. This increment is the smallest price movement the instrument can make, and may differ by instrument. For example, a tick on the e-mini S&P 500 (ES) equates to a movement of 0.25, while a tick on AAPL stock equates to a movement of 0.01. More information on setting an instrument's **Tick Size**can be found on the [Editing Instruments](https://ninjatrader.com/support/helpGuides/nt8/editing_instruments.htm) page. |       BarTypes3    1. Each historical bar in the 4 **Range** chart shown below represents exactly 4 ticks of price movement.  2. The **Ruler** [Drawing Tool](https://ninjatrader.com/support/helpGuides/nt8/working_with_drawing_tools__ob.htm) verifies that each bar consists of 4 ticks. (The "Y value" of 1.00 shown in the **Ruler's** display flag is equivalent to 4 ticks for the e-mini S&P 500 continuous contract instrument on the chart.) |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?bar_types.htm#UnderstandingRangeBars)

tog_minus        [Understanding Time based bars](javascript:HMToggle('toggle','UnderstandingTimeBasedBars','UnderstandingTimeBasedBars_ICON'))

|  |  |
| --- | --- |
| **Time Bars**  **Second**, **Minute**, **Day**, **Week**, **Month**, and **Year** bars are all built based on the passage of time. A bar will develop for a specified amount of time, and once this time is exceeded, a new bar will begin.    BarTypes4    1. Each historical bar in the 1 **Minute** chart shown below represents price movement during one minute in time.  2. The "Bar Timer" [indicator](https://ninjatrader.com/support/helpGuides/nt8/working_with_indicators.htm) has been applied to the chart to show the time remaining for the current bar.     |  | | --- | | **Note**: Intraday time based charts are built off the [Trading Hours](https://ninjatrader.com/support/helpGuides/nt8/trading_hours.htm) definitions set for the individual chart's [DataSeries](https://ninjatrader.com/support/helpGuides/nt8/working_with_price_data.htm). For daily charts and higher this is not the case though, here the Trading Hours are governed by the provider recording the data. For the NinjaTrader Historical Data Servers daily bars will be recorded using the ETH (Electronic Trading Hours) definitions for the respective instrument. Further for providers which support accessing the official settlement value, NinjaTrader will use this value as the daily bar close - for more information regarding your specific provider please consult [this link](https://ninjatrader.com/support/helpGuides/nt8/data_by_provider.htm). | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?bar_types.htm#UnderstandingTimeBasedBars)

tog_minus        [Understanding Heiken Ashi bars](javascript:HMToggle('toggle','UnderstandingHeikenAshiBars','UnderstandingHeikenAshiBars_ICON'))

|  |  |
| --- | --- |
| **Heiken Ashi Bars**  **Heiken Ashi** in Japanese translates to "Average Bar" in English. These bars are intended as a way to isolate ongoing trends. **Heiken Ashi** bars may appear to plot the Open, High, Low, and Close of price within a specified time period, similar to Candlestick bars. However, these bars use unique formulas to calculate OHLC values based on mathematical averages. Like Candlesticks, **Heiken Ashi** bars are based on the passage of time, and can be set to any **Second**, **Minute**, **Day**, **Week**, **Month**, or **Year**interval.     |  | | --- | | **Note**:  Calculated value will be rounded to the instrument's nearest tick size.  This is done to ensure accuracy in order submission and execution during backtesting. |     The chart below displays **Heiken Ashi** bars based on a 2-minute interval:    BarTypes9 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?bar_types.htm#UnderstandingHeikenAshiBars)

tog_minus        [Understanding Kagi bars](javascript:HMToggle('toggle','UnderstandingKagiBars','UnderstandingKagiBars_ICON'))

|  |
| --- |
| **Kagi Bars**  **Kagi** bars are based on price movement. A **Kagi** bar will plot in the direction of price until price reverses a specified amount, known as the **Reversal**. The bar will then change direction, but stay the same color until the last bar's High or Low is surpassed. The length of time the bar will develop depends upon the **Base period**.    For example, suppose the price of an instrument is heading down and the **Reversal** is set to 2 ticks. The line will continue to plot downward until price reverses more than 2 ticks. At this point, the line will change direction, but stay red by default. Once the last **Kagi** bar High is exceeded, the line will change to green by default, and the same rules will apply in the opposite direction. The chart below displays a 1 Minute **Kagi** chart with a **Reversal** set to 2 ticks.    BarTypes5 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?bar_types.htm#UnderstandingKagiBars)

tog_minus        [Understanding Renko bars](javascript:HMToggle('toggle','UnderstandingRenkoBars','UnderstandingRenkoBars_ICON'))

|  |
| --- |
| **Renko Bars**  **Renko** bars are based on price movement. Each bar is known as a "brick," and is plotted as green by default when price is moving up and red by default when price is moving down. A new brick is plotted when price exceeds the High or Low of the previous brick by a specified amount, known as the **Brick size**. The chart below displays **Renko** bars with a **Brick size** of 7:    BarTypes6 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?bar_types.htm#UnderstandingRenkoBars)

tog_minus        [Understanding Point and Figure bars](javascript:HMToggle('toggle','UnderstandingPointAndFigureBars','UnderstandingPointAndFigureBars_ICON'))

|  |  |
| --- | --- |
| **PointAndFigure Bars**  **PointAndFigure** bars are built based on price movement. Each bar plots a column made up of either X's representing a rising price or O's representing a decreasing price. Each X or O is referred to as a "box" and represents the price distance defined by the **Box size** (set in terms of ticks). A new X or O box will be added to the bar when price moves more than the **Box size,**warranting the addition of another box.    Another parameter, called the **Reversal**, sets the amount of price movement needed from the High or Low to change from X's to O's, or from O's to X's. A column will continue indefinitely until a price reversal equal to the **Reversal** amount (set in number of boxes) occurs. There can never be two columns of X's or O's next to each other for a given session, as any additional X's or O's would be added to the current column instead. When a reversal occurs, the next column begins one box size above the last Low for X's, or one box size below the last High for O's.    For example, the chart below shows **PointAndFigure** bars based on a 1 Minute **Data Series**. The **Box size** is set to 4 and the **Reversal** is set to 3.     |  | | --- | | **Note**: The prices of the X's and O's are represented by the exact middle of the X or O, rather than the top or bottom. |     BarTypes7 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?bar_types.htm#UnderstandingPointAndFigureBars)

tog_minus        [Understanding Line Break bars](javascript:HMToggle('toggle','UnderstandingLineBreakBars','UnderstandingLineBreakBars_ICON'))

|  |
| --- |
| **Line Break Bars**  **Line Break** bars are built based on price movement. **Line Break** bars must break above or below the High or Low of a specific set of prior bars before a new bar will be drawn. The "Line Breaks" parameter sets the number of previous bars in the set whose High or Low the current price must break.    For example, if the "Line Breaks" parameter is set to 2, as shown in the chart below, the first bar will be drawn based on whether the Close was above or below the Open. The second bar in the chart is drawn with a green color by default if price exceeds the first bar's High and red by default if price drops below the Low of the first bar. No new bar is drawn if price does not exceed the High or Low of the previous bar. The third bar is only plotted once price breaks the High/Low of the last 2 bars, since the **LineBreaks** parameter is set to 2. If the last break occurred on the upside, a color change will occur when price breaks the last Low. If the last break occurred on the downside, a color change will occur when price breaks the last High.    BarTypes8 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?bar_types.htm#UnderstandingLineBreakBars)

tog_minus        [Understanding Order Flow Volumetric Bars](javascript:HMToggle('toggle','UnderstandingOrderFlowVolumetricBars','UnderstandingOrderFlowVolumetricBars_ICON'))

|  |
| --- |
| For information on how to work with the Order Flow Volumetric Bars and Bar Statistics, please see the [Order Flow Volumetric Bars](https://ninjatrader.com/support/helpGuides/nt8/order_flow_volumetric_bars.htm) page in the [OrderFlow +](https://ninjatrader.com/support/helpGuides/nt8/order_flow_plus.htm) section of the Help Guide. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?bar_types.htm#UnderstandingOrderFlowVolumetricBars)

tog_minus        [Understanding Order Flow Delta Bars](javascript:HMToggle('toggle','UnderstandingOrderFlowDeltaBars','UnderstandingOrderFlowDeltaBars_ICON'))

|  |
| --- |
| For information on how to work with the Order Flow Delta Bars, please see the [Order Flow Delta Bars](https://ninjatrader.com/support/helpGuides/nt8/order-flow-delta-bars.htm) page in the [OrderFlow +](https://ninjatrader.com/support/helpGuides/nt8/order_flow_plus.htm) section of the Help Guide. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?bar_types.htm#UnderstandingOrderFlowDeltaBars)

tog_minus        [Understanding Order Flow Price On Volume Bars](javascript:HMToggle('toggle','UnderstandingOrderFlowPriceOnVolumeBars','UnderstandingOrderFlowPriceOnVolumeBars_ICON'))

|  |
| --- |
| For information on how to work with the Order Flow Price On Volume  Bars, please see the [Order Flow Price On Volume Bars](https://ninjatrader.com/support/helpGuides/nt8/order-flow-price-on-volume-bars.htm) page in the [OrderFlow +](https://ninjatrader.com/support/helpGuides/nt8/order_flow_plus.htm) section of the Help Guide. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?bar_types.htm#UnderstandingOrderFlowPriceOnVolumeBars)

|  |  |
| --- | --- |
| **Navigation:**  [Release Notes](https://ninjatrader.com/support/helpGuides/nt8/release_notes.htm) > [8.0](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) >  **8.0.7.1** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/8_0_8_0.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/8_0_6_1.htm) |

**8.0.7.0 Release**

June 6, 2017

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature #** | **Status** | **Category** | **Comments** |
| 11562 | Added | Control Center | When the Control Center's width is reduced menu items now change to icons |
| 11597 | Added | Forex.com | Added support for account denomination mapping to CHF, JPY, USD, EUR, GBP |
| 11545 | Added | FX Board | Add ability to add instruments using the keyboard |
| 11549 | Added | Hot Key | Added additional Hot Key functions for tabs |
| 11672 | Added | NinjaScript Editor | Add Support For Visual Studio 2017 (requires 15.2 or later) |
| 11746 | Added | Regionalization | Added regionalization support for Portuguese and completed more Spanish regionalization |

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 11476 | Fixed | Alerts | In some scenarios Alerts wouldn't activate |
| 11580 | Fixed | ATI | Bars building from data received from LastPlayback with an API was building inconsistently |
| 11488 | Fixed | ATM Strategies | In some scenarios an 'unable to load ATM strategy template file' error could occur |
| 11745 | Fixed | ATM Strategies | Scaling out of an ATM did not properly modify order quantities |
| 11750 | Fixed | ATM Strategies | Large quantities could result in OCO not applying to all orders |
| 11717 | Fixed | ATM Strategies, CQG | In some scenarios an inflight execution could exit a position and send a close order |
| 11574 | Fixed | Bars, Chart | Error occurred when using a Default 24x7 daily chart with Brasilia timezone |
| 11603 | Fixed | Bars, Chart | Crosshair label was inconsistent with x-axis label on Brasilia time with Default 24x7 hours |
| 11724 | Fixed | Bars, Instruments | Forex data displayed zeros if tick size was adjusted |
| 11704 | Fixed | BarsType, Chart | Removed custom bars type could still be referenced by workspace |
| 11470 | Fixed | Chart | Canceling instrument switch with applied alert resulted in wrong instrument in drop down menu |
| 11514 | Fixed | Chart | Clicking search button on a chart searched for the data series |
| 11525 | Fixed | Chart | Zoom icon and zoom box could display unexpectedly in some scenarios |
| 11605 | Fixed | Chart | Copy and Paste menu items didn't display hot key |
| 11653 | Fixed | Chart | X-Axis time in the future plotted incorrectly on multi-time-frame chart with different period types |
| 11667 | Fixed | Chart | In some scenarios opening a second chart tab resulted in an error |
| 11694 | Fixed | Chart | Sub-menu stayed open after item was selected if it was in an overflow panel |
| 11699 | Fixed | Chart | Importing minute data and generating day bars resulted in duplicate or incorrectly dated bars |
| 11725 | Fixed | Chart | At times clicking on a charts panel while switching tabs locked the chart image |
| 11727 | Fixed | Chart | At times selecting the y-axis while switching tabs results in an error |
| 11742 | Fixed | Chart | Removing series from multi-series chart could result in an error |
| 11747 | Fixed | Chart | Mouse left-click and drag worked unexpectedly when equidistant bar spacing was disabled |
| 11624 | Fixed | Chart, Drawing | Drawing tools did not work on 4th panel of chart if occupied by an indicator alone |
| 11592 | Fixed | Chart, DrawingTool | Global arrow line could be incorrectly rendered in non equidistant charts |
| 11526 | Fixed | Chart, NinjaScript | Chart Styles were not properly disposing |
| 11736 | Fixed | Chart, Playback | Scrolling with fixed scale, center price on scale, and Playback connected caused range and scale to change |
| 11627 | Fixed | Chart, Strategy Analyzer | DrawRegion could duplicate on  Strategy Analyzer charts |
| 11690 | Fixed | Chart | There was an error when changing the data series that had an indicator applied to it that was based off of another indicator |
| 11531 | Fixed | Commissions | Adding commissions to ambiguous instrument resulted in an exception |
| 11537 | Fixed | Connections | Crash occurred when disconnecting in some scenarios with multiple connections |
| 11633 | Fixed | Control Center | In some scenario menu items were skipped over when using arrow keys |
| 11657 | Fixed | Control Center | Category and Message columns of Log Grid could be removed and then couldn't be added back |
| 11503 | Fixed | Control Center | Filled Buys/Filled Sells column on Account grid was not counting partial fills |
| 11713 | Fixed | Control Center | Account Data Window did not save to workspace |
| 11601 | Fixed | Connections | Starting NinjaTrader with no internet connection resulted in a crash |
| 11582 | Fixed | Data | Downloading historical tick data for current day erased last hour of previous day. |
| 11618 | Fixed | Data Grids | Exporting data grids to CSV with values over 1000 was adding an invalid comma |
| 11500 | Added | Database | Added equities that were missing from the database |
| 11516 | Fixed | Drawing | On multi-series charts drawing object anchor points could adjust unexpectedly |
| 11616 | Fixed | Drawing | Merging/separating series on multi-series chart could render duplicate global draw objects |
| 11706 | Fixed | Drawing, NinjaScript | Resolved some errors related to drawing object rendering |
| 11631 | Fixed | Drawing, NinjaScript | RemoveDrawObject did not work when object was not on the price panel |
| 11613 | Fixed | DrawingObjects, NinjaScript | Resolved some scenarios of error failed to call OnRender() for chart object 'Ray'. "External component has thrown an exception." |
| 11102 | Fixed | DrawingTool | TrendChannel could display incorrectly in some scenarios where the anchor was off the chart |
| 11540 | Fixed | DrawingTool | Fibonacci Extension could place the price level label on the wrong side |
| 11589 | Fixed | DrawingTool | Ruler tick size format was wrong after switching instruments on chart |
| 11594 | Fixed | DrawingTool | Removed unused BackgroundOpacity property from RiskReward drawing tool |
| 11654 | Fixed | DrawingTool | Drawing object with future-anchors behaved poorly on multi-series chart |
| 11658 | Fixed | DrawingTool | VerticalLine selection points could be incorrectly distributed on additional chart panels |
| 11703 | Fixed | DrawingTool | Risk Reward tool with Auto scale enabled resulted in Y-axis to greatly extend |
| 11714 | Fixed | DrawingTool | When Gann fan's price levels were removed upon adding a new level a percentage field was given |
| 11721 | Fixed | DrawingTool | Restoring drawing heavy workspace could result in error |
| 11730 | Fixed | DrawingTool | Text objects moved after accessing the Drawing Objects window |
| 11412 | Fixed | DrawingTool, Strategy Analyzer | Memory could increase when reusing tag for a drawing object |
| 11669 | Fixed | eSignal | BANKNIFTY and NIFTY futures instruments were not able to receive data |
| 11524 | Fixed | Hot Key | Sell Stop limit order placed with hot key placed as simulated Sell Stop limit order |
| 11674 | Fixed | Hot Key | In some scenarios Hot Key order entry showed order confirmation twice |
| 11508 | Fixed | Indicator | There were missing spaces in TickCounter, VolumeCounter, and BarTimer |
| 11534 | Fixed | Indicator | Set ParabolicSAR to calculate on price change and resolved a case when its plot could back up from it's previous value |
| 11628 | Fixed | Indicator | Volume Profile with certain window sizes did not display latest profile as expected |
| 11632 | Fixed | Indicator | When adding BarTimer to a chart it briefly showed a disconnected message while still connected |
| 11655 | Fixed | Indicator | Swing Indicator was not outputting values for certain intervals |
| 11659 | Fixed | Indicator | BarTimer wouldn't function with simulated data feed if there were no historical bars |
| 11591 | Fixed | Interactive Brokers | In some scenarios switching instruments could remove execution markers |
| 11604 | Fixed | Interactive Brokers | Gateway connection stayed yellow when active orders/positions were present |
| 11607 | Fixed | Interactive Brokers | In some scenarios a working profit target/stop loss from Traders Workstation were not reported |
| 11619 | Fixed | Interactive Brokers | In a strategy shorting an instrument that that wasn't available to do so stopped the strategy and required manual cancellation of the order |
| 11615 | Fixed | IQFeed | Connection could stay in Connection Lost on failed log in attempt |
| 11553 | Fixed | Market Analyzer | Row highlight could not be disabled |
| 11567 | Fixed | Market Analyzer | Realized PnL for a simulation account reset inconsistently in different windows |
| 11590 | Fixed | Market Analyzer | Account drop-down was not populating all available accounts if the Market Analyzer was created before a data feed connection was started |
| 11662 | Fixed | Market Analyzer | Label row background color would reset when column background color changed |
| 11676 | Fixed | Market Analyzer, NinjaScript | Null references could result in a crash |
| 11568 | Fixed | Market Analyzer, Playback | In some scenarios adding an indicator column resulted in an error |
| 11732 | Fixed | Market Analyzer, SuperDOM | Indicator properties changed orientation in SuperDOM and Market Analyzer |
| 11504 | Fixed | NinjaScript | Enum could revert back to default parameter after compile |
| 11518 | Fixed | NinjaScript | SetProfitTarget as MIT order had limit price and was not modified correctly |
| 11555 | Fixed | NinjaScript | BarsSinceEntryExecution resulted in multi-series error on OrderFillResolution=High with empty overload |
| 11561 | Fixed | NinjaScript | Stop and target handling reverted to per entry execution on disable strategy |
| 11670 | Fixed | NinjaScript | Free trials produced an error requiring a restart for the trial to work |
| 11702 | Fixed | NinjaScript | Multi-series strategies using StartBehavior.AdoptAccountPosition will add the account's position on an instrument once per matching data series to the strategy's position |
| 11663 | Fixed | NinjaScript | Resolved some scenarios of error D2DERR\_WRONG\_FACTORY when rendering |
| 11737 | Fixed | NinjaScript | DrawRegion could be missing after hosting script saw OnBarUpdate() calls |
| 11761 | Fixed | NinjaScript | Resolved some scenarios of error Write lock may not be acquired with read lock held |
| 11661 | Fixed | NinjaScript | Resolved some scenarios of error on calling 'OnRender' method: Attempted to read or write protected memory |
| 11664 | Fixed | NinjaScript | Resolved some scenarios of error D2DERR\_WRONG\_STATE/WrongState, Message: The object was not in the correct state to process the method |
| 10173 | Fixed | NinjaScript Editor | XML Comments were not resolving correctly on DisplayAttribute properties |
| 10207 | Fixed | NinjaScript Editor | Was unable to properly collapse and expand regions holding only xml comments |
| 11535 | Fixed | NinjaScript Editor | Renaming of folders had unexpected results |
| 11626 | Added | NinjaScript Editor | Updated NinjaScript code editor |
| 11734 | Fixed | NinjaScript Editor | Right clicking NinjaScript Editor resulted in errors in Visual Studio output |
| 11550 | Fixed | NinjaScript, SharedAdapter | Twitter Share Service was not working with scripts |
| 11696 | Fixed | NinjaScript, Strategy Analyzer | Strategy indicator input series was defaulting to primary series when 'Optimize Data Series' was enabled in an optimization |
| 11630 | Fixed | NinjaScript, Templates | Saving a template for an indicator prevented changes to the plot names in code from taking affect |
| 11600 | Fixed | Playback | UTC  Amsterdam +1 time zone could result in a lockup |
| 11640 | Fixed | Playback | Recorded playback data was not time stamped correctly if disconnects occurred |
| 10869 | Fixed | Playback, Tick Replay | Reloading historical data with multiple strategies on multiple tick replay charts could result in an error |
| 11611 | Fixed | Strategy | In some scenarios the input series of the strategy instance on the tab was blank |
| 11510 | Fixed | Strategy | In some scenarios editing a strategies start behavior results in error and prevented connection |
| 11566 | Fixed | Strategy | In some scenarios an error occurred when adding a strategy to the strategies tab |
| 11665 | Fixed | Strategy | Account name displayed differently in strategies than other areas |
| 11556 | Fixed | Strategy | Error occurred when opening strategies dialog on chart with running strategy after removing source |
| 11413 | Fixed | Strategy Analyzer | Memory was not releasing when adding an indicator that draws objects |
| 11522 | Fixed | Strategy Analyzer | Optimization could hang on invalid high order resolution period value |
| 11523 | Fixed | Strategy Analyzer | In some scenarios running a test on an instrument with the option aggregated checked resulted in a crash |
| 11560 | Fixed | Strategy Analyzer | In some scenarios incorrect strategy name could show in drop-down menu |
| 11625 | Fixed | Strategy Analyzer | In some scenarios running tests with non-equidistant bar spacing resulted in an error |
| 11505 | Fixed | Strategy Builder | Removing a condition tab and compiling did not reflect the change in the code until relaunching Strategy Builder |
| 11506 | Fixed | Strategy Builder | In some scenarios a script error could make the Strategy Builder and NinjaScript editor inaccessible |
| 11507 | Fixed | Strategy Builder | Share to email resulted in an error |
| 11512 | Fixed | Strategy Builder | Calling an indicator in a sub-folder resulted in an error |
| 11521 | Fixed | Strategy Builder | Spaces and special characters in the strategy name were able to be input |
| 11571 | Fixed | Strategy Builder | BarsSinceEntryExecution was not using Multiseries overloads |
| 11610 | Fixed | Strategy Builder | Offset series barsAgo was not accounted for in CurrentBars check |
| 11652 | Fixed | Strategy Builder | Could not check if a Date series is equal to a date |
| 11570 | Fixed | SuperDOM | Could not change bar type for an indicator |
| 11520 | Fixed | SuperDOM | Sending Futures instrument from Market Analyzer to Dynamic SuperDOM resulted in an error |
| 11448 | Fixed | TD AMERITRADE | Incorrect order type was used to close position |
| 11637 | Fixed | TD AMERITRADE | Out of order events could result in canceled orders showing as working |
| 11643 | Fixed | TD AMERITRADE | Incorrect error displayed if 'user is not allowed to access Streamer' |
| 11666 | Fixed | Templates | Tick Replay was saved in templates which was invalid when Tick Replay was disabled |
| 11595 | Fixed | Trade Performance | Individual trades calculated in pips showed in tenth pips |
| 11698 | Fixed | Trade Performance | Orders and Executions did not populate orders if dates were set to current day |
| 11543 | Fixed | UI | Modal Windows were globally 'Always On Top' instead of Application level 'Always On Top' |
| 11623 | Fixed | UI | Mouse pointer did not change to a caret for End Date setting |
| 11697 | Fixed | UI, Drawing Obejects | Drawing object's anchor points didn't use PC's region format |
| 11701 | Done | Yahoo | Yahoo discontinued |

**8.0.7.1 Release**

June 21, 2017

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 11793 | Fixed | Playback | Having 'Playback current day only' disabled and moving the slider was only playing 1 day |
| **Navigation:**  [Release Notes](https://ninjatrader.com/support/helpGuides/nt8/release_notes.htm) > [8.0](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) >  **8.0.7.1** | | | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/8_0_8_0.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/8_0_6_1.htm) |

**8.0.7.0 Release**

June 6, 2017

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature #** | **Status** | **Category** | **Comments** |
| 11562 | Added | Control Center | When the Control Center's width is reduced menu items now change to icons |
| 11597 | Added | Forex.com | Added support for account denomination mapping to CHF, JPY, USD, EUR, GBP |
| 11545 | Added | FX Board | Add ability to add instruments using the keyboard |
| 11549 | Added | Hot Key | Added additional Hot Key functions for tabs |
| 11672 | Added | NinjaScript Editor | Add Support For Visual Studio 2017 (requires 15.2 or later) |
| 11746 | Added | Regionalization | Added regionalization support for Portuguese and completed more Spanish regionalization |

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 11476 | Fixed | Alerts | In some scenarios Alerts wouldn't activate |
| 11580 | Fixed | ATI | Bars building from data received from LastPlayback with an API was building inconsistently |
| 11488 | Fixed | ATM Strategies | In some scenarios an 'unable to load ATM strategy template file' error could occur |
| 11745 | Fixed | ATM Strategies | Scaling out of an ATM did not properly modify order quantities |
| 11750 | Fixed | ATM Strategies | Large quantities could result in OCO not applying to all orders |
| 11717 | Fixed | ATM Strategies, CQG | In some scenarios an inflight execution could exit a position and send a close order |
| 11574 | Fixed | Bars, Chart | Error occurred when using a Default 24x7 daily chart with Brasilia timezone |
| 11603 | Fixed | Bars, Chart | Crosshair label was inconsistent with x-axis label on Brasilia time with Default 24x7 hours |
| 11724 | Fixed | Bars, Instruments | Forex data displayed zeros if tick size was adjusted |
| 11704 | Fixed | BarsType, Chart | Removed custom bars type could still be referenced by workspace |
| 11470 | Fixed | Chart | Canceling instrument switch with applied alert resulted in wrong instrument in drop down menu |
| 11514 | Fixed | Chart | Clicking search button on a chart searched for the data series |
| 11525 | Fixed | Chart | Zoom icon and zoom box could display unexpectedly in some scenarios |
| 11605 | Fixed | Chart | Copy and Paste menu items didn't display hot key |
| 11653 | Fixed | Chart | X-Axis time in the future plotted incorrectly on multi-time-frame chart with different period types |
| 11667 | Fixed | Chart | In some scenarios opening a second chart tab resulted in an error |
| 11694 | Fixed | Chart | Sub-menu stayed open after item was selected if it was in an overflow panel |
| 11699 | Fixed | Chart | Importing minute data and generating day bars resulted in duplicate or incorrectly dated bars |
| 11725 | Fixed | Chart | At times clicking on a charts panel while switching tabs locked the chart image |
| 11727 | Fixed | Chart | At times selecting the y-axis while switching tabs results in an error |
| 11742 | Fixed | Chart | Removing series from multi-series chart could result in an error |
| 11747 | Fixed | Chart | Mouse left-click and drag worked unexpectedly when equidistant bar spacing was disabled |
| 11624 | Fixed | Chart, Drawing | Drawing tools did not work on 4th panel of chart if occupied by an indicator alone |
| 11592 | Fixed | Chart, DrawingTool | Global arrow line could be incorrectly rendered in non equidistant charts |
| 11526 | Fixed | Chart, NinjaScript | Chart Styles were not properly disposing |
| 11736 | Fixed | Chart, Playback | Scrolling with fixed scale, center price on scale, and Playback connected caused range and scale to change |
| 11627 | Fixed | Chart, Strategy Analyzer | DrawRegion could duplicate on  Strategy Analyzer charts |
| 11690 | Fixed | Chart | There was an error when changing the data series that had an indicator applied to it that was based off of another indicator |
| 11531 | Fixed | Commissions | Adding commissions to ambiguous instrument resulted in an exception |
| 11537 | Fixed | Connections | Crash occurred when disconnecting in some scenarios with multiple connections |
| 11633 | Fixed | Control Center | In some scenario menu items were skipped over when using arrow keys |
| 11657 | Fixed | Control Center | Category and Message columns of Log Grid could be removed and then couldn't be added back |
| 11503 | Fixed | Control Center | Filled Buys/Filled Sells column on Account grid was not counting partial fills |
| 11713 | Fixed | Control Center | Account Data Window did not save to workspace |
| 11601 | Fixed | Connections | Starting NinjaTrader with no internet connection resulted in a crash |
| 11582 | Fixed | Data | Downloading historical tick data for current day erased last hour of previous day. |
| 11618 | Fixed | Data Grids | Exporting data grids to CSV with values over 1000 was adding an invalid comma |
| 11500 | Added | Database | Added equities that were missing from the database |
| 11516 | Fixed | Drawing | On multi-series charts drawing object anchor points could adjust unexpectedly |
| 11616 | Fixed | Drawing | Merging/separating series on multi-series chart could render duplicate global draw objects |
| 11706 | Fixed | Drawing, NinjaScript | Resolved some errors related to drawing object rendering |
| 11631 | Fixed | Drawing, NinjaScript | RemoveDrawObject did not work when object was not on the price panel |
| 11613 | Fixed | DrawingObjects, NinjaScript | Resolved some scenarios of error failed to call OnRender() for chart object 'Ray'. "External component has thrown an exception." |
| 11102 | Fixed | DrawingTool | TrendChannel could display incorrectly in some scenarios where the anchor was off the chart |
| 11540 | Fixed | DrawingTool | Fibonacci Extension could place the price level label on the wrong side |
| 11589 | Fixed | DrawingTool | Ruler tick size format was wrong after switching instruments on chart |
| 11594 | Fixed | DrawingTool | Removed unused BackgroundOpacity property from RiskReward drawing tool |
| 11654 | Fixed | DrawingTool | Drawing object with future-anchors behaved poorly on multi-series chart |
| 11658 | Fixed | DrawingTool | VerticalLine selection points could be incorrectly distributed on additional chart panels |
| 11703 | Fixed | DrawingTool | Risk Reward tool with Auto scale enabled resulted in Y-axis to greatly extend |
| 11714 | Fixed | DrawingTool | When Gann fan's price levels were removed upon adding a new level a percentage field was given |
| 11721 | Fixed | DrawingTool | Restoring drawing heavy workspace could result in error |
| 11730 | Fixed | DrawingTool | Text objects moved after accessing the Drawing Objects window |
| 11412 | Fixed | DrawingTool, Strategy Analyzer | Memory could increase when reusing tag for a drawing object |
| 11669 | Fixed | eSignal | BANKNIFTY and NIFTY futures instruments were not able to receive data |
| 11524 | Fixed | Hot Key | Sell Stop limit order placed with hot key placed as simulated Sell Stop limit order |
| 11674 | Fixed | Hot Key | In some scenarios Hot Key order entry showed order confirmation twice |
| 11508 | Fixed | Indicator | There were missing spaces in TickCounter, VolumeCounter, and BarTimer |
| 11534 | Fixed | Indicator | Set ParabolicSAR to calculate on price change and resolved a case when its plot could back up from it's previous value |
| 11628 | Fixed | Indicator | Volume Profile with certain window sizes did not display latest profile as expected |
| 11632 | Fixed | Indicator | When adding BarTimer to a chart it briefly showed a disconnected message while still connected |
| 11655 | Fixed | Indicator | Swing Indicator was not outputting values for certain intervals |
| 11659 | Fixed | Indicator | BarTimer wouldn't function with simulated data feed if there were no historical bars |
| 11591 | Fixed | Interactive Brokers | In some scenarios switching instruments could remove execution markers |
| 11604 | Fixed | Interactive Brokers | Gateway connection stayed yellow when active orders/positions were present |
| 11607 | Fixed | Interactive Brokers | In some scenarios a working profit target/stop loss from Traders Workstation were not reported |
| 11619 | Fixed | Interactive Brokers | In a strategy shorting an instrument that that wasn't available to do so stopped the strategy and required manual cancellation of the order |
| 11615 | Fixed | IQFeed | Connection could stay in Connection Lost on failed log in attempt |
| 11553 | Fixed | Market Analyzer | Row highlight could not be disabled |
| 11567 | Fixed | Market Analyzer | Realized PnL for a simulation account reset inconsistently in different windows |
| 11590 | Fixed | Market Analyzer | Account drop-down was not populating all available accounts if the Market Analyzer was created before a data feed connection was started |
| 11662 | Fixed | Market Analyzer | Label row background color would reset when column background color changed |
| 11676 | Fixed | Market Analyzer, NinjaScript | Null references could result in a crash |
| 11568 | Fixed | Market Analyzer, Playback | In some scenarios adding an indicator column resulted in an error |
| 11732 | Fixed | Market Analyzer, SuperDOM | Indicator properties changed orientation in SuperDOM and Market Analyzer |
| 11504 | Fixed | NinjaScript | Enum could revert back to default parameter after compile |
| 11518 | Fixed | NinjaScript | SetProfitTarget as MIT order had limit price and was not modified correctly |
| 11555 | Fixed | NinjaScript | BarsSinceEntryExecution resulted in multi-series error on OrderFillResolution=High with empty overload |
| 11561 | Fixed | NinjaScript | Stop and target handling reverted to per entry execution on disable strategy |
| 11670 | Fixed | NinjaScript | Free trials produced an error requiring a restart for the trial to work |
| 11702 | Fixed | NinjaScript | Multi-series strategies using StartBehavior.AdoptAccountPosition will add the account's position on an instrument once per matching data series to the strategy's position |
| 11663 | Fixed | NinjaScript | Resolved some scenarios of error D2DERR\_WRONG\_FACTORY when rendering |
| 11737 | Fixed | NinjaScript | DrawRegion could be missing after hosting script saw OnBarUpdate() calls |
| 11761 | Fixed | NinjaScript | Resolved some scenarios of error Write lock may not be acquired with read lock held |
| 11661 | Fixed | NinjaScript | Resolved some scenarios of error on calling 'OnRender' method: Attempted to read or write protected memory |
| 11664 | Fixed | NinjaScript | Resolved some scenarios of error D2DERR\_WRONG\_STATE/WrongState, Message: The object was not in the correct state to process the method |
| 10173 | Fixed | NinjaScript Editor | XML Comments were not resolving correctly on DisplayAttribute properties |
| 10207 | Fixed | NinjaScript Editor | Was unable to properly collapse and expand regions holding only xml comments |
| 11535 | Fixed | NinjaScript Editor | Renaming of folders had unexpected results |
| 11626 | Added | NinjaScript Editor | Updated NinjaScript code editor |
| 11734 | Fixed | NinjaScript Editor | Right clicking NinjaScript Editor resulted in errors in Visual Studio output |
| 11550 | Fixed | NinjaScript, SharedAdapter | Twitter Share Service was not working with scripts |
| 11696 | Fixed | NinjaScript, Strategy Analyzer | Strategy indicator input series was defaulting to primary series when 'Optimize Data Series' was enabled in an optimization |
| 11630 | Fixed | NinjaScript, Templates | Saving a template for an indicator prevented changes to the plot names in code from taking affect |
| 11600 | Fixed | Playback | UTC  Amsterdam +1 time zone could result in a lockup |
| 11640 | Fixed | Playback | Recorded playback data was not time stamped correctly if disconnects occurred |
| 10869 | Fixed | Playback, Tick Replay | Reloading historical data with multiple strategies on multiple tick replay charts could result in an error |
| 11611 | Fixed | Strategy | In some scenarios the input series of the strategy instance on the tab was blank |
| 11510 | Fixed | Strategy | In some scenarios editing a strategies start behavior results in error and prevented connection |
| 11566 | Fixed | Strategy | In some scenarios an error occurred when adding a strategy to the strategies tab |
| 11665 | Fixed | Strategy | Account name displayed differently in strategies than other areas |
| 11556 | Fixed | Strategy | Error occurred when opening strategies dialog on chart with running strategy after removing source |
| 11413 | Fixed | Strategy Analyzer | Memory was not releasing when adding an indicator that draws objects |
| 11522 | Fixed | Strategy Analyzer | Optimization could hang on invalid high order resolution period value |
| 11523 | Fixed | Strategy Analyzer | In some scenarios running a test on an instrument with the option aggregated checked resulted in a crash |
| 11560 | Fixed | Strategy Analyzer | In some scenarios incorrect strategy name could show in drop-down menu |
| 11625 | Fixed | Strategy Analyzer | In some scenarios running tests with non-equidistant bar spacing resulted in an error |
| 11505 | Fixed | Strategy Builder | Removing a condition tab and compiling did not reflect the change in the code until relaunching Strategy Builder |
| 11506 | Fixed | Strategy Builder | In some scenarios a script error could make the Strategy Builder and NinjaScript editor inaccessible |
| 11507 | Fixed | Strategy Builder | Share to email resulted in an error |
| 11512 | Fixed | Strategy Builder | Calling an indicator in a sub-folder resulted in an error |
| 11521 | Fixed | Strategy Builder | Spaces and special characters in the strategy name were able to be input |
| 11571 | Fixed | Strategy Builder | BarsSinceEntryExecution was not using Multiseries overloads |
| 11610 | Fixed | Strategy Builder | Offset series barsAgo was not accounted for in CurrentBars check |
| 11652 | Fixed | Strategy Builder | Could not check if a Date series is equal to a date |
| 11570 | Fixed | SuperDOM | Could not change bar type for an indicator |
| 11520 | Fixed | SuperDOM | Sending Futures instrument from Market Analyzer to Dynamic SuperDOM resulted in an error |
| 11448 | Fixed | TD AMERITRADE | Incorrect order type was used to close position |
| 11637 | Fixed | TD AMERITRADE | Out of order events could result in canceled orders showing as working |
| 11643 | Fixed | TD AMERITRADE | Incorrect error displayed if 'user is not allowed to access Streamer' |
| 11666 | Fixed | Templates | Tick Replay was saved in templates which was invalid when Tick Replay was disabled |
| 11595 | Fixed | Trade Performance | Individual trades calculated in pips showed in tenth pips |
| 11698 | Fixed | Trade Performance | Orders and Executions did not populate orders if dates were set to current day |
| 11543 | Fixed | UI | Modal Windows were globally 'Always On Top' instead of Application level 'Always On Top' |
| 11623 | Fixed | UI | Mouse pointer did not change to a caret for End Date setting |
| 11697 | Fixed | UI, Drawing Obejects | Drawing object's anchor points didn't use PC's region format |
| 11701 | Done | Yahoo | Yahoo discontinued |

**8.0.7.1 Release**

June 21, 2017

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 11793 | Fixed | Playback | Having 'Playback current day only' disabled and moving the slider was only playing 1 day |
| **Navigation:**  [Operations](https://ninjatrader.com/support/helpGuides/nt8/operations.htm) > [Charts](https://ninjatrader.com/support/helpGuides/nt8/charts.htm) >  **Working with Multiple Data Series** | | | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/working_with_price_data.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/charts.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/bar_types.htm) |

**Multiple Data Series**

Multiple **Data Series** objects can be be viewed within a single chart window, and there are several ways to add **Data Series** to a chart.

tog_minus        [How to Add a Data Series](javascript:HMToggle('toggle','HowToAddADataSeries','HowToAddADataSeries_ICON'))

|  |
| --- |
| **Adding Data Series**  When you open a new chart, one or more **Data Series** will be applied, based on the instruments that you selected when creating the chart. You can add more **Data Series** to the chart (or remove existing **Data Series**) at any time via the following process:    1. Open the **Data Series** dialogue by either clicking the **Data Series** menu item on the chart toolbar, right-clicking in the chart and selecting the **Data Series** menu item from the Right Click menu, or double-clicking any selected **Data Series** on the chart.  2. Use the **Instrument Selector** or the **Search Tool** above the "Applied" section in the **Data Series** window to select a new **Data Series**.  3. Configure the **Data Series'** parameters as desired in the "Properties" section, then click the **OK** button    AddDataSeries    In the image above, we can use the **Instrument Selector** to add a **Data Series** to a chart which is already open. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_multiple_data_series.htm#HowToAddADataSeries)

tog_minus        [Managing Multiple Data Series](javascript:HMToggle('toggle','ManagingMultipleDataSeries','ManagingMultipleDataSeries_ICON'))

|  |  |
| --- | --- |
| **Multiple Data Series**  The image below shows two Data Series plotted within one chart window:    1.ES ##-## (1 Min)  2.AAPL (150 Tick)    AddDataSeries2    Each instrument is placed in its own panel by default, with the scale shown in the right margin of the chart. Many separate panels can be displayed within a single chart window. Instruments and indicators can alternatively be plotted within a single panel, as well. The scale of each **Data Series** can be justified to the right, to the left, or overlayed on the panel. Please see the "*Understanding panels*" section of the [Navigating a Chart](https://ninjatrader.com/support/helpGuides/nt8/navigating_a_chart.htm) page for more information.     |  | | --- | | **Tip**: When more than one panel is displayed in a chart, you can temporarily maximize a panel to fill the entire chart window by right mouse clicking in the price axis of that specific panel, then selecting the **Maximize** option. To restore the panel to its original size and placement, you can then right mouse click in the price axis of the maximized panel, then select the **Restore** option. |     **Equidistant Bar Spacing**  Equidistant Bar Spacing is a chart property that determines whether bars are plotted with an equal distance from each other or plotted on a horizontal axis with even time spacing. The two images below display the same chart with this property set to True and False. When set to True, the distance between bars is equal throughout the chart. When set to false, the distance between bars is not necessarily the same. Bars are instead plotted on a fixed x-axis timeline on which every inch along the axis represents an equal amount of time. This provides the benefit of being able to gauge momentum on non-time based charts, such as tick or volume, by visualizing how long it takes to finish building the next bar. Gaps may occur if no bar formed during the time interval, and overlapping bars may occur if bars are formed near the same time period. Both gaps and overlapping can be seen in the second image below. Equidistant Bar Spacing can be enabled or disabled within the [Chart Properties window](https://ninjatrader.com/support/helpGuides/nt8/chart_properties.htm).    AddDataSeries3ad    The image above shows two 150 **Tick** **Data Series** with "Equidistant Bar Spacing" set to True.    AddDataSeries4    The image above shows the same two **Data Series** with "Equidistant Bar Spacing" set to False.    **Equidistant Bar Spacing with Multiple Data Series**  When adding two or more **Data Series** to a chart, the bar spacing will be determine by the "Primary" data series, which is typically the first series added to the chart.    **Configuring Which Data Series is Primary**  You can optionally re-configure another series to be "Primary" by  right clicking on the chart bars and selecting "**Set as Primary**".    **Aggregated X-Axis Time Line**  When using multiple **Data Series** with different **Trading Hours** templates, NinjaTrader will set the time axis scale using the earliest begin time and latest end time of all **Trading Hours** templates applied to the **Data Series** on the chart. For example, if one instrument has a session begin time of 7:00 AM and an end time of 2:00 PM, and another has a session begin time of 8:00 AM and an end time of 4:00 PM, the chart will have a session begin time of 7:00 AM (from the first instrument) and an end time of 4:00 PM (from the second instrument). |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_multiple_data_series.htm#ManagingMultipleDataSeries)

|  |  |
| --- | --- |
| **Navigation:**  [Operations](https://ninjatrader.com/support/helpGuides/nt8/operations.htm) > [Order Entry](https://ninjatrader.com/support/helpGuides/nt8/order_entry.htm) >  **Working With Forex** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/fifo_optimization.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/order_entry.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/where_do_your_orders_reside_.htm) |

NinjaTrader supports trading and viewing market data for spot forex pairs, in addition to other supported instrument types. Due to the unique nature of forex markets, there are a number of features throughout the platform tailored specifically to these instruments, and a few considerations to keep in mind when working with forex in NinjaTrader.

tog_minus        [Pips Calculation Mode](javascript:HMToggle('toggle','PipsCalculationMode','PipsCalculationMode_ICON'))

|  |
| --- |
| **Pips vs. Ticks**  The "Pips" Calculation Mode can be used to calculate PnL and performance metrics throughout the platform. This mode allows you to tailor performance reporting specifically to your forex trades. Similar to the "Ticks" mode, "Pips" takes the lowest granularity of price movement for a forex instrument (called a tick in NinjaTrader), then divides it by 10 to arrive at the pip value for the instrument. For example, when viewing a USD/JPY quote of 113.67'5, the "7" would be the pip value, and the "5" would be the tick. Using the Pips Calculation Mode, the number of ticks in profit (the "5" in the example) will be divided by 10 to arrive at the number of pips of profit or loss.    Forex1    **Setting the Pips Calculation Mode**  The Pips calculation mode can be used in realized/unrealized PnL fields in trading windows ([Chart Trader](https://ninjatrader.com/support/helpGuides/nt8/chart_trader.htm), [SuperDOM](https://ninjatrader.com/support/helpGuides/nt8/superdom.htm), [Basic Entry](https://ninjatrader.com/support/helpGuides/nt8/basic_entry.htm), etc.), the [Trade Performance](https://ninjatrader.com/support/helpGuides/nt8/trade_performance.htm) window, and the [Strategy Analyzer](https://ninjatrader.com/support/helpGuides/nt8/strategy_analyzer.htm). In Trading Windows, the calculation mode can be changed by left-clicking within the PnL field, or by opening the window's Properties dialogue. For more information, see the relevant pages for each trading window.    Forex2    In the Trade Performance window and Strategy Analyzer, the calculation mode can be changed via the **Display** dropdown menu, which affects all relevant statistics.    Forex3 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_forex.htm#PipsCalculationMode)

tog_minus        [Pips in ATM Strategies](javascript:HMToggle('toggle','PipsInATMStrategies','PipsInATMStrategies_ICON'))

|  |  |
| --- | --- |
| **ATM Strategy Parameters**  The **Parameter Type** field within the ATM Strategy Parameters window can be changed to "Pips" to affect the way that stop loss and profit target prices are set by an ATM strategy. Just like the Pips PnL calculation mode, the Pips parameter type is based on a multiplicative factor of the Ticks parameter type (1 Pip = 10 Ticks). For example, rather than entering 200 ticks for your profit target (200 ticks = 20 pips), you can simply specify 20 pips.    Forex4     |  | | --- | | **Note**: If your forex data provider supports tenth-pip quotes, then you can also use the Ticks parameter type to set ATM orders with a sub-pip granularity. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_forex.htm#PipsInATMStrategies)

tog_minus        [Forex Lot Sizes](javascript:HMToggle('toggle','ForexLotSizes','ForexLotSizes_ICON'))

|  |  |
| --- | --- |
| **Setting Your FX Lot Size**  A "Forex Lot Size" property can be set for accounts shown in the Accounts tab of the Control Center. This setting affects the default position size populated in trading windows when a forex instrument is selected. To access this property, first select the Accounts  tab in the Control Center. Next, right click on the account you wish to edit, and select the **Edit Account** menu item. In the window that appears, set the Forex Lot Size property to your desired value. You can enter any amount here, whether or not it corresponds to a standard position size (Lot, Mini-Lot, Micro-Lot). For example, you could enter "102000" to automatically use a position size equal to one standard lot (100,000) plus two micro lots (2,000).    Forex6     |  | | --- | | **Notes**:  •The Forex Lot Size property does not prevent you from entering or selecting different position sizes in trading windows, but only controls what is populated in the Quantity field by default. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_forex.htm#ForexLotSizes)

tog_minus        [Forex-Specific Trading Windows](javascript:HMToggle('toggle','ForexSpecificTradingWindows','ForexSpecificTradingWindows_ICON'))

|  |
| --- |
| **FX Pro**  The FX Pro window is laid out similarly to the Basic Entry window, with a few enhancements and modifications tailored specifically to forex instruments. For more information on using this window, see the [FX Pro](https://ninjatrader.com/support/helpGuides/nt8/fx_pro.htm) page.    **FX Board**  The FX Board is a unique forex trading window featuring a grid of two-sided tiles updated in real time, offering market data, spread info, and order management functionality for multiple pairs at once. FX Pro and FX Board windows can be linked together via [Instrument Linking](https://ninjatrader.com/support/helpGuides/nt8/linking_windows.htm). When linked, you can simply click any tile in the FX Board, and the corresponding instrument will be selected in a linked FX Pro window. For more information on using this window, see the [FX Board](https://ninjatrader.com/support/helpGuides/nt8/fx_board.htm) page.    **Other Windows**  Forex instruments can be traded in other windows, as well, and are not limited to the two mentioned above. Forex-specific windows can also be linked to others via Instrument Linking. Other windows, such as Chart Trader or the Market Analyzer, do not include forex-specific features, but are capable of handling FX instruments just like any others. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_forex.htm#ForexSpecificTradingWindows)

tog_minus        [How Bars Are Built and Orders Filled](javascript:HMToggle('toggle','HowBarsAreBuiltAndOrdersFilled','HowBarsAreBuiltAndOrdersFilled_ICON'))

|  |  |
| --- | --- |
| **Building Bars with "Last Price" Data Type**  Forex price quotes do not use the concept of "Last Price" the same as other markets; only Bid and Ask quotes are available. Thus, when building bars using the default "Last" price type, the Bid price will be used instead. Using this price type, all bars on a chart will be built using Bid price updates, but you can choose to use the Ask price instead, if you wish. To change the price type used, first open the Data Series window on a chart, then toggle the value in the "Price Based On" field to your desired type.    **Realtime Order Fills vs. Backtesting**  Due to the absence of a last traded price quote in forex, all Buy orders in a live market are filled at the Ask price, and all Sell orders are filled at the Bid. However, when backtesting NinjaScript strategies, all simulated order fills will occur at the Bid price, regardless of whether they were Buy or Sells orders    Forex5    1.Ask Price: All realtime Buy orders are filled at the Ask  2.Bid Price: All realtime Sell orders and all backtest Buys and Sells are filled at the Bid     |  | | --- | | **Note**: In backtesting, a slippage value can be set to recreate the impact of the Bid/Ask spread on trade profit and loss. NinjaScript developers can calculate the spread in strategy logic, then dynamically set the [Slippage](https://ninjatrader.com/support/helpGuides/nt8/slippage.htm) property before entering orders. For non-programmers, an estimated slippage value can be applied to all trades via the Backtest/Optimization Properties section in the [Strategy Analyzer](https://ninjatrader.com/support/helpGuides/nt8/strategy_analyzer.htm). | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_forex.htm#HowBarsAreBuiltAndOrdersFilled)

tog_minus        [Forex Trading Hours](javascript:HMToggle('toggle','ForexTradingHours','ForexTradingHours_ICON'))

|  |
| --- |
| **Forex Trading Hours Template**  All forex instruments are configured to use the pre-defined "Forex" Trading Hours template, which runs 24 hours per day from 5:00pm EST on Sunday to 5:00pm EST on Friday, with an End-of-Day session break at 5:00pm each day. This covers the full range of forex trading throughout the week, but other Trading Hours templates can be applied to restrict the data on your charts to be in line with any local market timing on which you may wish to focus. For more information, see the [Trading Hours](https://ninjatrader.com/support/helpGuides/nt8/trading_hours.htm) page.    forex7 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_forex.htm#ForexTradingHours)

tog_minus        [Calculating Pip Value](javascript:HMToggle('toggle','CalculatingPipValue','CalculatingPipValue_ICON'))

|  |  |
| --- | --- |
| **How to Calculate the Pip Value for a Forex Pair**  Multiplying the pip size of your currency pair by the lot size of your order will provide you the pip value. This will be in the quote/counter currency of the forex pair. The quote/counter currency is the second currency in the pair.    **Example in USD for an USD Quote/Counter Currency**  In the following example we will do this for a 10,000 lot on the EURUSD. The quote/counter currency is USD and the EURUSD's point size is 0.0001.  10,000 x 0.0001 = 1  This indicates that 1 pip would be $1 USD.    CalculatigPip_USD    **Example in USD for a GBP Base Currency**  In the following example we will use the EURGBP. Let's say our account is in USD and we want to convert the pip value to USD. Again we will say we are trading a 10,000 lot size and the pip value for the EURGBP is 0.0001.  10,000 x 0.0001 = 1  This indicates that 1 pip would be £1 GBP. We would then multiply this by what the GBPUSD is trading at.  In this example the GBPUSD is trading at $1.26 (rounded).  1 x 1.26 = 1.26  This indicates that 1 pip would be $1.26 USD.    CalculatigPip_GBP       |  | | --- | | **Note**: If the conversion rate is not available the PnL information will be in the counter/quote currency of the pair. This would create a discrepancy in your Trade Performance. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?working_with_forex.htm#CalculatingPipValue)

|  |  |
| --- | --- |
| **Navigation:**  [Release Notes](https://ninjatrader.com/support/helpGuides/nt8/release_notes.htm) > [8.0](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) >  **8.0.4.0 Release Notes** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/8_0_5_2.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/8_0_3_0.htm) |

**Release Date**

January 31, 2017

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 11056 | Fixed | Adapter | In some cases quickly disconnecting and reconnecting resulted in an error |
| 10989 | Fixed | Alerts | In some scenarios disabled alerts would re-enable unexpectedly when re-opening the workspace |
| 11027 | Fixed | Alerts, Drawing Tool | Alerts were not detecting trend line cross overs properly |
| 11091 | Fixed | ATM Strategies | Submitting an order to an active ATM strategy for an opposite position left an active strategy |
| 11028 | Fixed | Backup & Restore | Drawing object templates were not transferred when restoring from a backup |
| 10855 | Fixed | Chart | Using the 'Apply' button on the strategies dialog did not always perform the requested action. |
| 10964 | Fixed | Chart | Y-axis crosshair label was not equal to value in the y-axis scale when two data series where on a chart |
| 10971 | Fixed | Chart | In specific data load requests for tick data the full range requested did not display in the chart |
| 11002 | Fixed | Chart | Non-equidistant bar spacing caused invalid x-axis labels on interval change |
| 11011 | Fixed | Chart | Center on price scale with fixed scale incorrectly adjusted on each tick instead of on each new bar |
| 11035 | Fixed | Chart | Second data series was included in auto scale calculations despite auto scale being disabled |
| 11060 | Fixed | Chart | Region did not render when equidistant bar spacing was set to false. |
| 11061 | Fixed | Chart | Region and plot could get out of sync when using a displacement |
| 11064 | Fixed | Chart | Region and plot display did not sync with right side margin |
| 11071 | Fixed | Chart | When making setting changes the chart could lose your scroll location and reset back to displaying the last bar on chart. |
| 11078 | Fixed | Chart | Changing instrument via right click menu resulted in a blank chart |
| 11098 | Fixed | Chart | Adding Line On Close to a regular trading hours data series caused an error |
| 11076 | Fixed | Chart, DrawingTool | Region with displacement and non-equidistant bar spacing caused incorrect display |
| 11073 | Fixed | Chart, Hot Key | Databox was not closing with hot key |
| 11054 | Fixed | Chart, NinjaScript | In some scenarios moving an indicator that places drawing objects resulted in an error |
| 10995 | Fixed | Chart, SuperDOM | Editing an instrument applied to a chart and SuperDOM could make NinjaTrader unresponsive |
| 11070 | Fixed | Chart, Workspaces | Left fixed scale setting was not saving |
| 11047 | Fixed | Code Wizard | Resolved an error when trying to edit an added Custom Data Series |
| 11100 | Fixed | Commissions | Entering Commission Per Instrument Type in template caused an error |
| 11032 | Fixed | CQG, Backup & Restore | Performing a Backup/Restore after you had been connected to CQG to not finish correctly |
| 10789 | Fixed | Drawing Tool | In some scenarios the Trend Channels area wouldn't fully fill |
| 10865 | Fixed | Drawing Tool | Moving drawing objects on a logarithmic chart scale could incorrectly place anchor points |
| 11031 | Fixed | Drawing Tool | Trend Channel line color would not change |
| 11038 | Fixed | Drawing Tool | Regression channel disappeared when Price Type was set to 'Median' |
| 11059 | Fixed | Drawing Tool | Region displacement was not properly displacing |
| 10968 | Fixed | DrawingTool, Indicator | Drawing Tool anchors would break when switching intervals with indicator on chart |
| 10895 | Fixed | eSignal | Was not able to reconnecting after manual disconnect |
| 10993 | Fixed | FXCM | In some scenarios stacked orders could produce an error |
| 11053 | Fixed | FXCM | Daily Charts wouldn't receive real-time data with FXCM historical data server |
| 11081 | Fixed | FXCM | Historical data from FXCM's servers showed volume as 100k |
| 11052 | Fixed | Hot Key | Chart Trader Hot keys required Chart Trader to be selected |
| 10958 | Changed | Indicator | Updated existing indicators and help guide to consistently advise and perform instantiation of Indicators and custom DataSeries in State.DataLoaded |
| 11016 | Fixed | Instruments | Instrument Lists were not updating on first time start up |
| 10947 | Fixed | Interactive Brokers | A bad bar was generated when left connected past regular market hours |
| 11068 | Fixed | Interactive Brokers | In some scenarios connection hung in yellow state when there were active orders |
| 11039 | Fixed | Interactive Brokers, Kinetick | In some scenarios current daily bar was removed when using Preferred Connections |
| 11093 | Fixed | IQFeed | In some scenarios IQFeed could request too many simultaneous historical data requests |
| 11009 | Fixed | Market Analyzer | Zeros were reporting in indicator column for PriorDayOHLC |
| 10917 | Fixed | NinjaScript | Strategy stopped trading after historical data was reloaded |
| 11017 | Changed | NinjaScript | AddChartIndicator was not working in State.DataLoaded |
| 10922 | Fixed | NinjaScript, ATM Strategies | Closing position via AtmStrategyClose method prevented RealizedPnL from updating correctly |
| 11051 | Fixed | Orders, Rithmic, FXCM | Long account information could prevent orders from updating |
| 10977 | Fixed | Performance | Using Window's Taskbar preview used large amount of resources |
| 10950 | Fixed | Playback | Controller became disabled/frozen when using Tick Replay |
| 11024 | Fixed | Playback, SuperDOM | Market Depth disappeared/reappeared when scrolling price ladder |
| 10961 | Fixed | Strategy | Moving chart tab while strategy was loading caused an exception |
| 11092 | Fixed | Strategy | High order fill resolution with multi-series script would disable rather than enable with popup message |
| 10767 | Fixed | Strategy Analyzer | In some scenarios the chart was missing plot lines and the chart became transparent |
| 10966 | Fixed | Strategy Analyzer | In some scenarios the walk forward optimization results were not matching the summary page |
| 10978 | Fixed | Strategy Analyzer | Time frame date was wrong when opening at the start of a new year |
| 10992 | Fixed | Strategy Analyzer | Could not change grid row with arrow up and down keys |
| 11004 | Fixed | Strategy Analyzer | Show Tabs value was not consistently checked/unchecked |
| 11021 | Fixed | Strategy Analyzer | Changing from Summary to Trades display caused strategy to re-run |
| 11055 | Fixed | Strategy Analyzer | Changing optimization parameters could leads to an error |
| 11005 | Fixed | Strategy Builder | Selecting Close Price could display back as Default Input |
| 11006 | Fixed | Strategy Builder | Condition set tab order does not persist in Strategy Builder |
| 11007 | Fixed | Strategy Builder | Offset could not use User Defined Input |
| 11048 | Fixed | Strategy Builder | Defining a condition in Strategy Builder which was supposed to 'divide' a numerical value didn't actually perform the action |
| 11077 | Fixed | Strategy Builder | Default plot was not available for MACD |
| 10970 | Changed | Strategy Builder, Code Wizard | Added Use Primary Instrument option for adding data series |
| 11030 | Fixed | Strategy, Chart | In some scenarios strategy was not removed as underlying series was removed |
| 10983 | Fixed | SuperDOM | Center button was not working when last price was near top/bottom of ladder |
| 10951 | Fixed | TD Ameritrade, Indicator | In some scenarios the Bar Timer didn't properly function |
| 11044 | Fixed | Trading Hours | Eurex trading hours templates included Martin Luther King holiday |
| 11033 | Fixed | UI | Error could occur when using the Vendor Licensing Add On |
| 10991 | Fixed | Window Linking | In some scenarios changing time frames on a linked chart caused the current bar to report in different parameters |
| 10915 | Fixed | Workspaces | Added Skin Selection as part of installer |
| **Navigation:**  [Release Notes](https://ninjatrader.com/support/helpGuides/nt8/release_notes.htm) > [8.0](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) >  **8.0.20.1** | | | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/8_0_21_1.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/8_0_19_1.htm) |

**8.0.20.1 Release Date**

December 23, 2019

|  |
| --- |
| **Attention TD Ameritrade Users:**  •For TD Ameritrade uses there is a new connection process to authorize the username and password. To continue to connect you must update to NinjaTrader 8.0.20.0 or newer by December 31, 2019. See the link [here](https://ninjatrader.com/Advisories/TD-Ameritrade-Upgrade-NT8) for more information |

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue#** | **Status** | **Category** | **Comments** |
| 14246 | Fixed | TD Ameritrade, Orders | Session is now renewed every hour to ensure order submission works as expected |

**8.0.20.0 Release Date**

December 5, 2019

|  |
| --- |
| **Features** |
| **Added options support for Kinetick, IQFeed, and Interactive Brokers (beta**)  Kinetick, IQFeed, Interactive Brokers, Options  Feature #13676    With Kinetick, IQFeed, and Interactive Brokers you can now access options on futures and equities. At this time these features are in beta.    r2013676 |
| **Major improvements on Portuguese translations and sound files**  Localization  Feature #14185    Added improvements to Portuguese translations provide a more complete experience for Portuguese clients. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue#** | **Status** | **Category** | **Comments** |
| 14083 | Fixed | Alerts | Clicking an Alert Dialog box caused the box to move off screen |
| 14113 | Fixed | Alerts | Alerts Log did not save column settings |
| 14014 | Fixed | Alerts, Chart | Corrupt alert could prevent other alerts on the same chart from working |
| 14090 | Fixed | ATI | CLOSESTRATEGY command did not close orders or position from Atm Strategy |
| 13981 | Fixed | ATM Strategies | Target Chase activated without the target being touched |
| 14002 | Done | ATM Strategies | Reverse at stop/target orders would get canceled on disconnect and get stuck on cancel submitted |
| 14129 | Fixed | ATM Strategies | ATM strategy template was not saving when Chase if Touched was applied/taken off |
| 14164 | Fixed | ATM Strategies | An ATM with 'MIT enabled for Profit' would submit the MIT target order with a stop price of zero if position was added to an active ATM strategy |
| 14135 | Changed | Backup & Restore | Improved handling of references for backup/restore process |
| 14009 | Fixed | Chart | Resolved a scenario where cross hair cursor icon was incorrect after restarting |
| 14045 | Fixed | Chart | Dragging a data series from one chart to another did not keep bar width from originating chart |
| 14055 | Fixed | Chart | X axis label was missing on minute chart with some scaling settings |
| 14073 | Fixed | Chart | Resolved an error that could occur when the data box was saved off screen in a workspace |
| 14077 | Fixed | Chart | 'No time scroll' cross hair setting did not display associated hot key |
| 14133 | Fixed | Chart | Chart templates would grow in size from saving information unrelated to the template |
| 14038 | Fixed | Chart, Window Linking, NinjaScript | Duplicate in new window while using instrument link could lose track of panel index |
| 14062 | Fixed | Control Center | Removing three DLLs which were applied to a chart would cause a crash |
| 14152 | Fixed | Control Center, Commissions | Executions tab commission column did not honor account denomination property |
| 14027 | Fixed | Core | Failed to process install file error now auto repairs so it should only occur once |
| 14123 | Added | CQG | Added Open Interest with WebAPI |
| 14142 | Fixed | CQG | WebAPI would not display real time index data |
| 14145 | Fixed | CQG | Resolved a scenario where some WebAPI orders would be accepted at a different price |
| 14146 | Fixed | CQG | WebAPI would not attempt to reconnect after a lost connection |
| 14018 | Fixed | Depth Chart | Property presets were not applied at start-up |
| 14101 | Fixed | DrawingTool | Resetting a drawing object template caused drawing object to disappear |
| 14125 | Fixed | DrawingTool | Moving the ruler tool outside of chart boundaries resulted in a loop of errors |
| 14138 | Fixed | FXCM | Resolved a scenario where connection loss could get stuck |
| 14081 | Fixed | Historical Data Window | Some import errors did not display in the format expected |
| 14036 | Fixed | Hot Key, Market Watch | Market Watch hot key was not listed under Global Hot Keys |
| 14102 | Fixed | Interactive Brokers | Bars with a volume of zero would show with a volume of one |
| 14160 | Fixed | Interactive Brokers | Resolved a scenario where orders would return to an incorrect contract month that does not exist |
| 14149 | Fixed | Interactive Brokers | Orders submitted at an invalid forex prices did not throw order rejection/error |
| 13966 | Fixed | Interactive Brokers, NinjaScript | Stop order was unexpectedly canceled when disabling a strategy |
| 14050 | Fixed | Interactive Brokers, Position Display | VX futures positions did not show up on order entry position display |
| 14119 | Fixed | Licensing | 3rd Party Licensing incorrectly accepted user defined IDs with spaces |
| 14187 | Fixed | Licensing | Vendor licensing filter by Name returned no results |
| 14115 | Fixed | Localization, Commissions | Commissions Dialog Instrument Type was not localized |
| 13967 | Fixed | Market Analyzer | Resolved a scenario where configured indicator name didn't update as expected |
| 14065 | Fixed | Market Analyzer | Cross above/below cell conditions did not work as expected |
| 14127 | Fixed | Market Analyzer | Bar graph percent did not properly span full column width |
| 14139 | Fixed | Market Analyzer | Bar graph did not show negative values |
| 14140 | Fixed | Market Analyzer | Bar graph tooltip value was not rounded |
| 14060 | Fixed | Market Analyzer, Indicator | Correlation indicator produced error when applied |
| 13965 | Fixed | NinjaScript | Resolved a scenario where canceling a stop order from a strategy resulted in an error |
| 14067 | Fixed | NinjaScript | Switching tabs could lose cursor position |
| 14163 | Fixed | NinjaScript | Resolved a scenario where a DLL included in a backup file did not get added to Bin/Custom on restore |
| 13969 | Fixed | NinjaScript | Updated SampleOnOrderUpdate to track all execution and ensure the fills match before submitting stop market and limit orders |
| 14112 | Fixed | NinjaScript Editor | Importing NinjaScript file will caused focus of NinjaScript tab to change |
| 14170 | Fixed | NinjaScript Editor | Find window remained visible when changing workspaces |
| 13996 | Fixed | NinjaScript, ATM Strategies | GetAtmStrategyRealizedProfitLoss sometimes reported 0 after AtmStrategyClose was used |
| 14053 | Fixed | NinjaScript, Chart | Global draw objects placed by script plotted incorrectly on lower time frame charts |
| 13848 | Fixed | Option Chain | 'Loading...' could stay stuck after disconnect |
| 14097 | Fixed | Options Chain | Middle ruler  jumped around as new market data came in |
| 14084 | Fixed | Order Flow + | Order Flow Volume Profile POC and Value Area lines extended outside of trading hours |
| 14104 | Fixed | Order Flow + | Order Flow Volume Profile price profile could plot letters from outside of trading hours |
| 14105 | Fixed | Order Flow + | Order Flow Volume Profile price boxes overlapped when profile alignment was set to right |
| 14108 | Fixed | Order Flow + | Order Flow Volume Profile could cause chart to lag when 'Display mode' was set to 'Outline' |
| 14167 | Fixed | Order Flow + | Order Flow Volume Profile composite profile trading hours property did not affect the profile |
| 14171 | Fixed | Order Flow + | Order Flow Volume Profile Initial Balance accounted for 1 extra bar of data |
| 14192 | Fixed | Order Flow + | Order Flow Volumetric bars did not apply gradient to statistic values in some scenarios |
| 14173 | Fixed | Performance | Recent instruments could retain unnecessary information |
| 14012 | Changed | Performance, Chart | Improved performance for 1440 minute charts |
| 14041 | Fixed | Performance, Chart | Unchecking 'Equidistant bar spacing' could result in lag |
| 14054 | Fixed | Playback | Start and End Dates did not save when disconnecting and reconnecting |
| 14068 | Fixed | Playback | A multi-series strategy behaved differently if applied on the strategies tab vs a chart |
| 14103 | Fixed | Playback, NinjaScript | ImmediatelySubmit was submitting duplicate orders in Playback |
| 14154 | Fixed | Playback, NinjaScript | Strategy using CloseStrategy() caused a freeze |
| 14087 | Fixed | Playback, Strategy, Trade Performance | Error occurred when viewing real-time trade performance in Playback |
| 14150 | Fixed | Position Display | Submitting an order to an invalid instrument resulted in a crash |
| 14162 | Fixed | Stock Import | Could not import stocks to create instruments with numbers |
| 13972 | Fixed | Strategy Analyzer | Opening AI Generate Strategy Analyzer result in new Strategy Analyzer resulted in an error |
| 13976 | Fixed | Strategy Analyzer | Parameters tool-tip in the Log showed the parameters for the wrong test |
| 13990 | Fixed | Strategy Analyzer | Double clicking a Walk Forward result in the Log resulted in the Order, Execution, and Trades displays do not populating |
| 13998 | Fixed | Strategy Analyzer | Performance value in Results grid did not match value in summary during optimization |
| 14032 | Fixed | Strategy Analyzer | Backtests with a high order fill resolution of 1 tick resulted in an error |
| 14051 | Fixed | Strategy Analyzer | Monte Carlo report for MaxConsecutiveWinners and MaxConsecutiveLosers were reversed |
| 14120 | Fixed | Strategy Analyzer | Optimizer could get an error when running off historical data with some sets of data and settings |
| 14124 | Fixed | Strategy Analyzer | Compiling reset some parameter settings |
| 14148 | Fixed | Strategy Analyzer | Backtest with a commission template applied and 'Display' set to 'Percent' incorrectly formatted commission value after a restart |
| 14151 | Fixed | Strategy Analyzer | Resetting strategy template then running a backtest changed parameter values after backtest |
| 14046 | Fixed | Strategy Builder | Using an indicator as an input resulted in a compile error |
| 14143 | Fixed | Strategy Builder | Order Flow + indicators were incorrectly available in the Strategy Builder |
| 14179 | Fixed | Strategy Builder | User defined inputs would allow quotation marks when it should not |
| 14181 | Fixed | Strategy Builder | Creating an action which sets custom series to its value 1 bar ago would change after a restart and reopening of the strategy in Strategy Builder |
| 14166 | Fixed | Strategy Builder | Using system indicator names for user inputs was incorrectly allowed |
| 14147 | Fixed | Strategy, Chart | Unchecking enabled on a strategy applied to chart, then removing the strategy without OK/Apply caused chart to retain strategy |
| 14080 | Fixed | SuperDOM | Dynamic SuperDOM could not select ATM drop down reliably |
| 14121 | Fixed | SuperDOM | Resolved a scenario where there could be a bars ago error when loading an indicator |
| 14033 | Fixed | SuperDOM, NinjaScript | Indicator that adds a daily series would  not load |
| 14111 | Fixed | SuperDOM, Window Linking | Window linking passed through to other workspaces when 'Global link button across workspaces' was disabled |
| 14064 | Fixed | Tick Replay, Strategy | Enabling a strategy with Tick Replay after a restart resulted in a crash |
| 14093 | Fixed | Tick Replay, Bars | Resolved a scenario where some scripts saved in a workspace with Tick Replay resulted in an error |
| 14074 | Fixed | Workspaces | DataBox was left off screen when opening workspace and clicking yes to move windows to primary monitor |

|  |  |
| --- | --- |
| **Navigation:**  [Release Notes](https://ninjatrader.com/support/helpGuides/nt8/release_notes.htm) > [8.0](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) >  **8.0.6.1** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/8_0_7_1.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/8_0.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/8_0_5_2.htm) |

**8.0.6.0 Release**

April 17, 2017

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature #** | **Status** | **Category** | **Comments** |
| 11311 | Added | FOREX.com | FOREX.com adapter added in (Beta) |
| 10617 | Changed | DrawingTool | Updated the visual style of various Drawing Tools and Trade Markers |
| 11427 | Added | IQFeed, Kinetick | 'VWAP' Fundamental Data support was added for IQFeed and Kinetick |
| 11459 | Added | Rithmic | Added support for Account 'Unrealized PnL' reported from Rithmic displayed in the 'Account Data' window. |
| 11444 | Changed | Trade Performance | The 'Trade Performance' window now filter in real-time, no longer requiring the report to be regenerated. |
| 10941 | Added | General | Added caption bar context menu for window operations (Restore, Minimize, Maximize, Close). Additionally when a window is maximized you can now drag the caption bar to restore the window. |
| 11452 | Changed | General | Reduced the height of tabs. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 11421 | Fixed | Alerts | Multiple share actions on a single alert attempting to take a screen shot would cause an error |
| 11451 | Fixed | Alerts | Resolved a few scenarios where Alert condition did not fire as expected |
| 11477 | Fixed | ATM Strategies | ATM selection mode settings prevented REV button from working |
| 11430 | Fixed | ATM Strategies | Scaling in with an ATM using Pip mode resulted in unexpected rounding |
| 11469 | Fixed | ATM Strategies, FX Board | Canceling out of setting up a new 'Custom' ATM strategy didn't revert to previous ATM strategy selection |
| 11332 | Fixed | ATM Strategies, Strategy | Having an active ATM and then enabling a NinjaScript strategy could result in an error |
| 11375 | Fixed | ATM Strategies | Orders tab ATM name didn't show full active name until reconnect |
| 11355 | Fixed | Bars, Tick Replay | GetBar had error with secondary series when Tick Replay was enabled |
| 11369 | Fixed | Basic Entry | Action column wasn't showing order state colors |
| 11467 | Fixed | Basic Entry, Chart Trader, FX Pro, SuperDOM | Some areas were transparent in dark skins |
| 11229 | Fixed | Chart | Couldn't scroll non-equidistant chart when chart bars building too quickly |
| 11313 | Fixed | Chart | Strategy could keep plot executions on chart as finalized by auto close |
| 11399 | Fixed | Chart | Displaced indicators showed values in Data Box without displacement |
| 11446 | Fixed | Chart | Auto Scale on Indicator panel was not working the same as Data Series panel |
| 11475 | Fixed | Chart | Some times zone settings could result in errors |
| 11275 | Fixed | Chart | Daily bars took longer to display than expected |
| 11339 | Fixed | Chart Trader | Resolved some race conditions that could prevent an OCO order from canceling |
| 11324 | Fixed | Chart, DrawingTool | Pasting text from one instrument to a different instrument prevented it's properties from being available |
| 11376 | Fixed | Continuum, CQG | In some scenarios race conditions could prevent OCO orders from canceling |
| 11321 | Fixed | Continuum, CQG | Daily volume for current day was higher than expected |
| 11335 | Fixed | Control Center | In some scenarios multi-instrument entry prices would not showing in strategies tab |
| 11315 | Fixed | Core | Resolved some in-flight-executions scenarios that could prevent OCO orders from canceling |
| 11447 | Fixed | Data Grids | Live accounts were not showing under 'Filter by accounts' |
| 11359 | Fixed | Data Grids | Order Grid text alignment was off after editing |
| 11445 | Fixed | Database | In some scenarios continuous contracts would show in instrument rollovers |
| 11278 | Fixed | DrawingTool | Draw objects could shrink in size when moved into the left chart boundary |
| 11364 | Fixed | DrawingTool | Tag names would increase per object rather than per object type |
| 11479 | Fixed | DrawingTool | Drawing objects would not persist through rollover when Attach To All Charts was selected |
| 11286 | Fixed | DrawingTool, NinjaScript | Re-using drawing tool tag with different type did not replace as expected |
| 11358 | Fixed | FX Board | Order grid incorrectly updated when adding targets |
| 11397 | Fixed | FX Board | Up and down movement arrows were not displaying as expected |
| 11366 | Fixed | FX Board, FX Pro | Adjusting the quantity with the arrows was not modifying by the account set FX lot sizes |
| 11485 | Fixed | Hot Key, Orders | Enabling OCO orders would not apply OCO to Hot Key orders |
| 11487 | Fixed | Hot Key, Orders | Enabling simulated orders would not simulate Hot Key orders |
| 11385 | Fixed | Indicator | Woodies CCI panel was different between NinjaTrader 7 & 8 |
| 11443 | Fixed | Indicator | Pivots Performance Improved |
| 11462 | Fixed | Indicator | BarTimer on daily chart wasn't showing message that it needs to be on intraday data |
| 11316 | Fixed | Instruments | Resolved some scenarios that could prevent instruments from showing in search windows |
| 11382 | Fixed | IQFeed | Bid, Ask, and Last historical data was displaying the same values |
| 11455 | Fixed | IQFeed | With some set ups and time zones not all requested data would load |
| 11379 | Fixed | Market Analyzer | PositionAvgPrice column wasn't showing tenth pip values |
| 11429 | Fixed | Market Analyzer | Changing the font removed Column background colors |
| 11415 | Fixed | Market Analyzer | Resizing columns was not working as expected when resizing window |
| 11344 | Fixed | NinjaScript | WaitUntilFlat only waited for the first exit of a position before submitting orders |
| 11368 | Fixed | NinjaScript | Resolved some scenarios of the error "Upgradeable lock may not be acquired with read lock held" |
| 11395 | Fixed | NinjaScript | Renaming script sub folder did not take affect until restart |
| 11424 | Fixed | NinjaScript | Resolved some scenarios of error "Failed to call OnRender() for chart object 'Line': 'External component has thrown an exception.'" |
| 11326 | Fixed | NinjaScript Editor | Improper focus could prevent seeing compile option |
| 11348 | Fixed | NinjaScript Editor | NinjaScript editor right click compile option was enabled inconsistently |
| 11293 | Done | NinjaScript, Tick Replay | Tick Replay and Order Fill Resolution = High was able to be used in combination |
| 11367 | Fixed | Orders | Updating order quantity of OCO orders failed to fully persist OCO |
| 11459 | Fixed | Orders | In some scenarios properly cancelled orders could show error 'order can not be cancelled' |
| 11391 | Fixed | Other | Automatic Windows Update of video card drivers could result in an error |
| 11340 | Fixed | Playback | Using playback data from daylight savings time could result in an error |
| 11438 | Fixed | Playback | Historical NinjaScript draw objects were not drawn when using the Playback Slider |
| 11203 | Fixed | Strategy | BarsTypes default overrides were not applying to strategies added to strategies grid of Control Center |
| 11418 | Fixed | Strategy | In some scenarios restoring strategies could add in duplicates / phantom instances |
| 11458 | Fixed | Strategy | Multi-series strategy applied wrong trading hours to second data series when enabled from strategies grid |
| 11482 | Fixed | Strategy | In some scenarios a strategy could be edited while still running |
| 11289 | Fixed | Strategy Analyzer | No count was returned in the Optimizer for 'this.Results[0].AllTrades.Count' |
| 11317 | Fixed | Strategy Analyzer | Selecting 'Open NinjaScript Output' was not functioning |
| 11319 | Fixed | Strategy Analyzer | Selecting 'Open result in New Strategy Analyzer' opened with the incorrect backtest type |
| 11386 | Fixed | Strategy Analyzer | Aggregate optimization caused identical Performance values for all instruments |
| 11387 | Fixed | Strategy Analyzer | Strategy box did not display strategy name if amended namespace |
| 11290 | Fixed | Strategy Builder | In some scenarios of multiple alerts with different rearm times only the shorter time would rearm |
| 11347 | Fixed | Strategy Builder | A strategy could be named with the same name as pre-existing indicator resulting in errors |
| 11361 | Fixed | SuperDOM, UI | At times changing the order quantity could display the wrong quantity in the confirmation window |
| 11448 | Fixed | TD AMERITRADE | Short positions were reported as a 'negative' value which throws off internal logic |
| 11330 | Fixed | Trade Performance | In some scenarios Average Trade could incorrectly show a value of zero |
| 11356 | Fixed | UI | Button Backgrounds for saved items where not updating when changing skins |
| 11406 | Fixed | UI | Chart's Taskbar button would not move with the chart when moved to different monitor |
| 11428 | Fixed | UI | In some scenarios Simulation Color would not update |
| 11349 | Fixed | Visual Studio Integration | File changes in Visual Studio was not triggering compile sound |
| 11383 | Fixed | Window Linking, Workspaces | In some scenarios Chart indicator panels reverted to default size when linked |

**8.0.6.1 Release**

April 25, 2017

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 11578 | Fixed | DrawingTools | Removed boxes around text in drawing objects and reverted minimum chart marker size implemented in feature 10617 |
| 11587 | Fixed | NinjaScript, DrawingTools | Bars ago was referencing BarsInProgress rather than the primary bars object |
| 11583 | Fixed | Continuum, CQG | Resolved some scenarios where an order couldn't be canceled until a reconnect occurred when using local OCO simulation |
| 11586 | Fixed | IQFeed | Daily data was displaying incorrect values |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) > [Educational Resources](https://ninjatrader.com/support/helpGuides/nt8/educational_resources.htm) > [Developing Indicators](https://ninjatrader.com/support/helpGuides/nt8/developing_indicators.htm) > [Advanced - Custom Plot Colors via Thresholds](https://ninjatrader.com/support/helpGuides/nt8/advanced_-_custom_plot_colors_.htm) >  **Entering Calculation Logic** | | | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/set_up8.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/advanced_-_custom_plot_colors_.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/compiling5.htm) |

The OnBarUpdate() method is called for each incoming tick or on the close of a bar (user defined) when performing real-time calculations and is called on each bar of a data series when re-calculating the indicator. For example, an indicator would be re-calculated when adding it to an existing chart that has existing price data displayed. Therefore, this is the main method called for indicator calculation and we will use this method to enter the script that will calculate the ROC value.

**Setting Plot Thresholds**  
The OnStateChange() method is called once before any bar data is loaded and is used to configure the indicator. The code below is automatically generated by the wizard and added to the OnStateChange() method. It configures the indicator for two plots and one line and sets the parameters.

| ns |
| --- |
| AddLine(Brushes.Black, 0, "ZeroLine"); AddPlot(Brushes.Green, "AboveZero"); AddPlot(Brushes.OrangeRed, "BelowZero"); |

Enter the following code in the OnStateChange() method and below the wizard generated code:

| ns |
| --- |
| // Set the threshold values for each plot Plots[0].Min = 0; Plots[1].Max = 0; |

The concept of setting threshold values is to set when and when not to paint a plot on the chart. In this indicator, we have an "AboveZero" plot with a default color of green which we only want to see when the value of ROC is above zero and a "BelowZero" plot with a default color of OrangeRed which we only want to see when the value of ROC is below zero. In order to make that happen we have to set the threshold values of each plot.

Plots[0].Min = 0;

This statement says, in the collection of Plot objects, take the first one (Plots[0]) and set its minimum value to zero. This means any value below zero will not display.

Plots[1].Max = 0;

This statement says, in the collection of Plot objects, take the second one (Plots[1]) and set its maximum value to zero. This means any value above zero will not display.

We now have a simple plot switching mechanism that displays the correct colored line depending on if the value of ROC is above or below zero. In fact, you can take this concept a little bit farther. You can even set different plots style (bar, dot etc..) depending on threshold values.

*A quick word about collections. Collections are objects that store a collection of objects, kind of like a container. In this case we are working with a collection of plots. In the above wizard generated code you will notice that we are adding new plots to the "Plots" collection. "AboveZero" was added first and then "BelowZero". This means that we can reference the "AboveZero" plot object through Plots[0]. The reason we don't pass in a value of 1 is because collections are zero based indexes. This means the first item has an index of 0, the second time an index of 1 and so forth.*

**Completing the Indicator**  
This indicator is actually quite simple in its implementation. The last thing we need to do is add the calculation code and set the value of ROC to both our plot lines.  
Replace the wizard generated code with the following code into the OnBarUpdate() method in the NinjaScript Editor:

| ns |
| --- |
| // Are there enough bars if (CurrentBar < Period) return;   // Set the plot values AboveZero[0] = SMA(ROC(Period), Smooth)[0]; BelowZero[0] = SMA(ROC(Period), Smooth)[0]; |

The calculation first checks to ensure there are enough bars to complete the calculation and then sets both plot lines to the ROC value.

The class code in your editor should look identical to the image below. You are now ready to [compile the indicator](https://ninjatrader.com/support/helpGuides/nt8/compiling5.htm) and configure it on a chart.

| ns | |
| --- | --- |
| public class CustomROC : Indicator {   protected override void OnStateChange()   {     if (State == State.SetDefaults)     {         Description                           = @"ROC with custom line color options";         Name                                 = "CustomROC";         Calculate                             = Calculate.OnBarClose;         IsOverlay                             = false;         DisplayInDataBox                     = true;         DrawOnPricePanel                     = true;         DrawHorizontalGridLines               = true;         DrawVerticalGridLines                 = true;         PaintPriceMarkers                     = true;         ScaleJustification                   = NinjaTrader.Gui.Chart.ScaleJustification.Right;         //Disable this property if your indicator requires custom values that cumulate with each new market data event.         //See Help Guide for additional information.         IsSuspendedWhileInactive             = true;         Period                               = 14;         Smooth                               = 3;         AddLine(Brushes.Black, 0, "ZeroLine");         AddPlot(Brushes.Green, "AboveZero");         AddPlot(Brushes.OrangeRed, "BelowZero");         Plots[0].Min = 0;         Plots[1].Max = 0;     }     else if (State == State.Configure)     {     }   }     protected override void OnBarUpdate()   {     // Are there enough bars     if (CurrentBar < Period) return;       // Set the plot values     AboveZero[0] = SMA(ROC(Period), Smooth)[0];     BelowZero[0] = SMA(ROC(Period), Smooth)[0];   }     #region Properties   [NinjaScriptProperty]   [Range(1, int.MaxValue)]   [Display(Name="Period", Description="Number of periods", Order=1, GroupName="Parameters")]   public int Period   { get; set; }     [NinjaScriptProperty]   [Range(1, int.MaxValue)]   [Display(Name="Smooth", Description="Smoothing rate", Order=2, GroupName="Parameters")]   public int Smooth   { get; set; }       [Browsable(false)]   [XmlIgnore]   public Series<double> AboveZero   {     get { return Values[0]; }   }     [Browsable(false)]   [XmlIgnore]   public Series<double> BelowZero   {     get { return Values[1]; }   }   #endregion } | |
| **Navigation:**  [Operations](https://ninjatrader.com/support/helpGuides/nt8/operations.htm) > [Charts](https://ninjatrader.com/support/helpGuides/nt8/charts.htm) >  **Working with Objects on Charts** | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/chart_panels.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/charts.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/working_with_price_data.htm) |

Charts in NinjaTrader can contain and display multiple objects, including **Data Series**, **Drawing Objects**, and indicator plots. Objects on charts can be managed in a number ways, such as dragging and dropping them to new panels, changing the axis of their price scale (if applicable), or changing the order in which they are painted on a chart.

tog_minus        [How to drag and drop chart objects](javascript:HMToggle('toggle','HowToDragAndDropChartObjects','HowToDragAndDropChartObjects_ICON'))

|  |
| --- |
| **Drag and Drop**  A **Data Series** or Indicator can be dragged and dropped to various areas of the chart to quickly change which panel it is displayed in.    Left mouse click on a chart object within a chart, then drag it to any of the following areas of the chart and release the mouse button:    1.Upper limit - Creates a new panel at the top of the chart  2.In between panels -  Creates a new panel in between two existing panels  3.Lower limit - Creates a new panel at the bottom of the chart  4.Center area of a panel - Relocates the selected chart object to this panel and automatically determines the most suitable scale justification  5.Left or right margin of a panel - Relocates the selected chart object to this panel (unless already in the selected panel) and changes the scale justification to the selected side of the panel.    When you drag a selected object to the upper or lower edge of a chart, or between two panels, a blue band will appear. This indicates that a new panel will be created when you drop the object in that location.    **Tabs and Windows**  In addition to moving around within a single chart tab, A **Data Series** or indicator can be dragged and dropped into any other chart window or tabs in your workspace. The following drag and drop actions can be performed:    1. Drag an indicator to an existing tab in any chart window - Duplicates the indicator in the tab or window into which it is dropped, leaving the original instance of the object intact  2. Drag a **Data Series** to an existing tab in any chart window - Replaces the primary **Data Series** in that tab with the one dropped into it  3. Drag a **Data Series** to the upper/lower limit, or between two panels, of a separate chart window -Creates a new panel, creating a multi-series chart if only one **Data Series** had previously been applied  2. Drag an indicator or **Data Series** to a New tab (+) - Creates a new tab and duplicates the object within it |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?chart_objects.htm#HowToDragAndDropChartObjects)

tog_minus        [How to copy and paste chart objects](javascript:HMToggle('toggle','HowToCopyAndPasteChartObjects','HowToCopyAndPasteChartObjects_ICON'))

|  |  |
| --- | --- |
| **Copy and Paste**  A **Data Series**, indicator, or **Drawing Object** can be copied and pasted to various areas of a chart to quickly duplicate an object and its properties. Chart objects can be copied in one of two ways:    •Left mouse click the chart object to select it. Next, right mouse click the object, then click the **copy** menu item.  •Left mouse click the chart object to select it, then use the Windows default CTRL + C Hot Key    After copying, chart objects can be pasted into the following areas:    •Current chart window or tab - **Data Series**and indicators will be duplicated in a new panel. **Drawing Objects** will be pasted with a slight offset from the copied object's location.  •Separate chart window or tab - **Data Series** will be placed in a new panel within the chart window or tab in which it is dropped. Indicators will either be plotted in an existing panel or in a new panel, depending on the indicator's "Overlay" property. **Drawing Objects** will be placed in the same panel number as the one from which they are copied, if it is available.     |  | | --- | | **Note**: When an indicator is pasted from one chart to another, the indicator will use the same input series if it is applied to the chart into which the indicator is pasted. Otherwise, it will use the second chart's primary **Data Series**. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?chart_objects.htm#HowToCopyAndPasteChartObjects)

tog_minus        [How to change the z-order (paint order) of a chart object](javascript:HMToggle('toggle','HowToChangeTheZorderpaintOrderOfAChartObject','HowToChangeTheZorderpaintOrderOfAChartObject_ICON'))

|  |  |
| --- | --- |
| **Z-Order**  Objects within a panel can be adjusted to appear behind or in front of another chart object. The specific layer on which an object sits is referred to as the "**z-order**."    You can change the **z-order** (paint order) of all chart objects within each individual panel. Each chart object is assigned a **z-order** value, which informs you where in the paint order that particular object resides. As a rule of thumb, there are as many **z-order** levels in a panel as there are chart objects in that panel. For example, if you had a **Data Series** and an SMA indicator in the same panel, there would be two painting levels. Level 1 is the top most level, which means that any chart object on Level 1 will be painted above all others. Continuing our example, if the **Data Series** was on Level 1 of 2 and the SMA indicator was on Level 2 of 2, that would mean the **Data Series** would be painted on top of the indicator.    The image below depicts a "Rectangle" drawing object set at z-order Level 3 of 3, which is behind both the Stochastics indicator (Level 2 of 3) and the ES ##-## Data Series (Level 1 of 3).    ChartObjects1    To adjust the **z-order** of an object:    1.Select the chart object by left mouse clicking on it  2.Hold down the "Shift" key on your keyboard and roll the mouse scroll wheel up or down to change the z-order of the object. As you scroll, the object's **z-order** will be displayed near your mouse cursor.     |  | | --- | | **Note**: **Drawing Objects** originating from a NinjaScript indicator or strategy will all generally share the same z-order as the script. In this case, the **z-order** of objects must be changed within the code of the indicator or strategy. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?chart_objects.htm#HowToChangeTheZorderpaintOrderOfAChartObject)

|  |  |
| --- | --- |
| **Navigation:**  [Operations](https://ninjatrader.com/support/helpGuides/nt8/operations.htm) > [Strategy Builder](https://ninjatrader.com/support/helpGuides/nt8/strategy_builder.htm) >  **Builder Screens** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/strategy_builder.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/strategy_builder.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/strategybuilder_condition_builder.htm) |

The Builder point and click interface is a powerful entrypoint into NinjaScript strategy development for non programmers. Even if you target more deeper custom coding later on in the development cycle, the Builder can provide a great foundation to start with. To get started directly into full fledged programming a strategy object in the NinjaScript editor, please check into [NinjaScript Wizard](https://ninjatrader.com/support/helpGuides/nt8/ns_wizard.htm).

|  |
| --- |
| playVideo |

tog_minus        [Understanding the Welcome screen](javascript:HMToggle('toggle','UnderstandingTheWelcomeScreen','UnderstandingTheWelcomeScreen_ICON'))

|  |
| --- |
| **Welcome Screen Layout**  This is the first screen and starting point in the **Strategy Builder**.    Strategy_Builder_1    1.In the **Strategy** drop-down select **New Strategy** to create a new strategy script - all other Builder made scripts will be listed as well, so should you wish to modify a script - please select the desired one and proceed through the screens.  2.Press the **View Code** button at any time to view the Builder generated NinjaScript code.  3.Press the **Unlock Code** button at any time to open the NinjaScript editor and edit your strategy code.  **Once the code is unlocked, you can no longer use the Builder for subsequent strategy editing**  4. Press the **Compile** button at any time to compile your strategy code.  5. Press the **<Back** or **Next>**buttons to move back or forth between Builder screens - you can also directly jump to a specific screen by using the left side navigation menu.  6. Press the **Cancel** button to leave the **Strategy Builder**    **Note:** Should you want to make a copy of your strategy, you can select your saved script in the **Strategy** drop-down and select '**save as**' - this opens a file dialog, where you can enter a new name to save the script copy under.    Strategy_Builder_2 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheWelcomeScreen)

tog_minus        [Understanding the General screen](javascript:HMToggle('toggle','UnderstandingTheGeneralScreen','UnderstandingTheGeneralScreen_ICON'))

|  |
| --- |
| **General Screen Layout**  The **General** screen is where you enter the name and description of your strategy.    Strategy_Builder_3    1. Sets the name of the strategy  2. Sets the description of the strategy |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheGeneralScreen)

tog_minus        [Understanding the Default properties screen](javascript:HMToggle('toggle','UnderstandingTheDefaultPropertiesScreen','UnderstandingTheDefaultPropertiesScreen_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Default properties screen Layout**  The Default properties screen is where you can set the default values for your custom strategy properties.    Strategy_Builder_4    1. Per default only the **Calculate** section is visible, click the **More properties** to expand the selection to include all strategy default properties as well to set for your Builder script.     |  |  | | --- | --- | | Calculate | Sets the **Calculation Mode** for the strategy. Possible values are "On Each Tick," "On Price Change," or "On Bar Close" | | Entries per direction | Sets the maximum number of entries allowed per direction while a position is active based on the "Entry handling" property | | Entry handling | Sets the manner in which entry orders are handled. If set to "AllEntries", the strategy will process all entry orders until the maximum allowable entries set by the "Entries per direction" property have been reached while in an open position. If set to "UniqueEntries", the strategy will process entry orders until the maximum allowable entries set by the "Entries per direction" property per each uniquely named entry have been reached. | | Exit on close | When enabled, open positions will be closed on the last bar of a session | | Exit on session close seconds | Sets the number of seconds prior to the end of a session at to close any open positions held by the strategy | | Fill Limit Orders on Touch | Enables the filling of limit orders when touched for the historical portion of the chart | | Maximum Bars Look Back | Sets the maximum number of historical bars to use for strategy calculations. The TwoHundredFiftySix setting is the most memory friendly | | Minimum Bars Required | Sets the minimum number of historical bars required to start taking trades | | Order Fill Resolution | Sets the way that simulated historical orders will be processed by the strategy. See the [Understanding Historical Fill Processing](https://ninjatrader.com/support/helpGuides/nt8/understanding_historical_fill_.htm) page for more information. | | Real-time error handling | Defines the behavior of a strategy when a strategy generated order is returning in a "Rejected" state. See the [Real-time Error Handling](https://ninjatrader.com/support/helpGuides/nt8/realtimeerrorhandling.htm) page for more information. | | Slippage | Sets the slippage amount in ticks for the historical portion of the chart | | Start Behavior | Sets the starting behavior of the strategy, based upon the account position. See the [Syncing Account Positions](https://ninjatrader.com/support/helpGuides/nt8/syncing_account_positions.htm) page for more information. | | Stops and Targets | Sets how stop and target orders are submitted | | Time in force | Sets the order's time in force. Possible values are DAY and GTC | | Trace orders | Enables sending more detailed order debug info to the NinjaScript [output window](https://ninjatrader.com/support/helpGuides/nt8/output.htm) | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheDefaultPropertiesScreen)

tog_minus        [Understanding the Additional data screen](javascript:HMToggle('toggle','UnderstandingTheAdditrionalDataScreen','UnderstandingTheAdditrionalDataScreen_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Additional data screen Layout**  The Additional data screen is where you can optionally select additional instrument data or custom series for your strategy.     |  | | --- | | **Critical**: You will want to make sure to add any additional series in this section that a hosted / called MultiSeries indicator in your Builder script would use, such as for example the [Pivots](https://ninjatrader.com/support/helpGuides/nt8/pivots.htm), [Camarilla Pivots](https://ninjatrader.com/support/helpGuides/nt8/camarilla_pivots.htm) or [Fibonacci Pivots](https://ninjatrader.com/support/helpGuides/nt8/fibonacci_pivots.htm) indicators. |     Strategy_Builder_5    1. Press the **add** button to be able to configure a new series to add  2. Press the **edit** button to be able to configure an existing series  3. Press the **remove** button to be able to remove an existing series    **Data Series Selector Layout**  Select your instrument data series to add here    Strategy_Builder_6     |  |  | | --- | --- | | Use primary instrument | Checking this will use the primary instrument the strategy is applied to | | Instrument | Select your instrument from the favorite or list selector or by using the search feature (press the magnifying glass) | | Price based on | Selects the price type the data series is based on, possible values are Last, Bid, Ask | | Type | Selects the bars type your series will use, possible values for the Builder interface are -    •Tick  •Minute  •Day  •Week  •Month  •Year  •Volume  •Range  •Second | | Value | Sets the bars period type value for your series |     **Custom Series Selector Layout**  Select your custom series to add here    Strategy_Builder_7   |  |  | | --- | --- | | Name | Set the name for your custom series | | Type | Selects the data type of the custom series, possible values for the Builder interface are -    •Bool  •Double  •DateTime  •Int  •String | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheAdditrionalDataScreen)

tog_minus        [Understanding the Inputs and Variables screen](javascript:HMToggle('toggle','UnderstandingTheInputsAndVariablesScreen','UnderstandingTheInputsAndVariablesScreen_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Inputs and Variables screen Layout**  The Inputs and Variables screen allows you to define the user inputs of your strategy. User inputs are important if you require input values that may vary the performance of your strategy. If for example you have a simple moving average cross over system, you may want to create an input for the fast moving average and another for the slow moving average. This then allows you to change the values of the moving averages at run time from the UI. Inputs are also required if you plan to use the NinjaTrader [Strategy Analyzer's optimization](https://ninjatrader.com/support/helpGuides/nt8/strategy_analyzer.htm) capabilities.    Strategy_Builder_8    1. Press the **add** button to add a new user input.  2. Press the **edit** button to edit an existing, selected user input.  3. Press the **remove** button to remove the selected user input.     |  |  | | --- | --- | | Name | Set the name for your user input | | Type | Selects the data type of the user input, possible values for the Builder interface are -    •Bool  •Double  •String  •Int  •Time | | Default | Set the default value your user input will have | | Min | Set the minimum value your user input will have | | Description | Enter an optional description for your user input here |     Strategy_Builder_9    1. Press the **add** button to add a new user variable.  2. Press the **edit** button to edit an existing, selected user variable.  3. Press the **remove** button to remove the selected user variable.     |  |  | | --- | --- | | Name | Set the name for your user variable | | Type | Selects the data type of the user variable, possible values for the Builder interface are -    •Bool  •Double  •String  •Int  •Time | | Default | Set the default value your user variable will have |      |  | | --- | | **Note**: If an input is named the same as the generated code for an indicator, the strategy will not be able to successfully compile. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheInputsAndVariablesScreen)

tog_minus        [Understanding the Conditions and Actions screen](javascript:HMToggle('toggle','UnderstandingTheConditionsAndActionsScreen','UnderstandingTheConditionsAndActionsScreen_ICON'))

|  |
| --- |
| **Conditions and Actions screen Layout**  The Conditions and Actions screen allows you to set conditions and subsequent actions that control the flow of your strategy.    *Conditions* - Take the specified action when true  *Actions* - Execute an action (submit orders, draw objects on the chart etc ...) based on its parent condition evaluating to true    Via the Builder, you can have an unlimited set of conditions with related actions and you also group conditions into a condition group (for example for a certain set of filter rules like time)    Conditions and condition groups are created using the [Condition Builder](https://ninjatrader.com/support/helpGuides/nt8/strategybuilder_condition_builder.htm). Actions are specified by the [Strategy Actions](https://ninjatrader.com/support/helpGuides/nt8/actions.htm) window.    Strategy_Builder_10    1. Selects **if all** of the individual conditions have to be met in order to trigger an action, or **if any** will be sufficient.  2. Displays the conditions associated with the currently selected condition set  3. Adds, opens condition grouping(\*), edits or removes a condition (a double click on selected item will also allow editing)  4. Displays the actions associated with the currently selected condition set  5. Adds, edits or removes an action (a double click on selected item will also allow editing)  6. Selects the condition set you wish to edit    \* For an example on working condition groups, please see "How to create a Time Filter' in the [Condition Builder](https://ninjatrader.com/support/helpGuides/nt8/strategybuilder_condition_builder.htm) section    You can copy and paste conditions from one set to another and you can even save a condition set as a template and load for future use via the right mouse button click context menu as show in the image below. To save a condition set as a template, select the **Save As...** menu item and then to re-use it in another strategy or condition set at a later time, select the**Load...** menu item.    Strategy_Builder_11 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheConditionsAndActionsScreen)

tog_minus        [Understanding the Stops and Targets screen](javascript:HMToggle('toggle','UnderstandingTheStopsAndTargetsScreen','UnderstandingTheStopsAndTargetsScreen_ICON'))

|  |
| --- |
| **Stops and Targets screen Layout**  The Stops and Targets allows you to set stop loss, trail stop, parabolic stop (R15 and higher) and profit target orders that are automatically submitted and managed once your strategy opens a position.    Strategy_Builder_12    1. Displays stops and targets associated with your strategy  2. Adds a stop or target to your strategy  3. Edits the selected stop or target in your strategy (a double click on the selected item will also allow editing)  4. Removes the selected stop or target from your strategy |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheStopsAndTargetsScreen)

tog_minus        [Understanding the Finish screen](javascript:HMToggle('toggle','UnderstandingTheFinishScreen','UnderstandingTheFinishScreen_ICON'))

|  |
| --- |
| **Finish screen Layout**  Once you reach this screen you are finished with developing your strategy. Press the **Finish** button to compile your strategy which will then be ready for [backtesting](https://ninjatrader.com/support/helpGuides/nt8/backtest_a_strategy.htm) or [live execution](https://ninjatrader.com/support/helpGuides/nt8/running_ninjascript_strategies.htm).    Strategy_Builder_13 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheFinishScreen)

|  |  |
| --- | --- |
| **Navigation:**  [Operations](https://ninjatrader.com/support/helpGuides/nt8/operations.htm) > [Strategy Builder](https://ninjatrader.com/support/helpGuides/nt8/strategy_builder.htm) >  **Builder Screens** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/strategy_builder.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/strategy_builder.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/strategybuilder_condition_builder.htm) |

The Builder point and click interface is a powerful entrypoint into NinjaScript strategy development for non programmers. Even if you target more deeper custom coding later on in the development cycle, the Builder can provide a great foundation to start with. To get started directly into full fledged programming a strategy object in the NinjaScript editor, please check into [NinjaScript Wizard](https://ninjatrader.com/support/helpGuides/nt8/ns_wizard.htm).

|  |
| --- |
| playVideo |

tog_minus        [Understanding the Welcome screen](javascript:HMToggle('toggle','UnderstandingTheWelcomeScreen','UnderstandingTheWelcomeScreen_ICON'))

|  |
| --- |
| **Welcome Screen Layout**  This is the first screen and starting point in the **Strategy Builder**.    Strategy_Builder_1    1.In the **Strategy** drop-down select **New Strategy** to create a new strategy script - all other Builder made scripts will be listed as well, so should you wish to modify a script - please select the desired one and proceed through the screens.  2.Press the **View Code** button at any time to view the Builder generated NinjaScript code.  3.Press the **Unlock Code** button at any time to open the NinjaScript editor and edit your strategy code.  **Once the code is unlocked, you can no longer use the Builder for subsequent strategy editing**  4. Press the **Compile** button at any time to compile your strategy code.  5. Press the **<Back** or **Next>**buttons to move back or forth between Builder screens - you can also directly jump to a specific screen by using the left side navigation menu.  6. Press the **Cancel** button to leave the **Strategy Builder**    **Note:** Should you want to make a copy of your strategy, you can select your saved script in the **Strategy** drop-down and select '**save as**' - this opens a file dialog, where you can enter a new name to save the script copy under.    Strategy_Builder_2 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheWelcomeScreen)

tog_minus        [Understanding the General screen](javascript:HMToggle('toggle','UnderstandingTheGeneralScreen','UnderstandingTheGeneralScreen_ICON'))

|  |
| --- |
| **General Screen Layout**  The **General** screen is where you enter the name and description of your strategy.    Strategy_Builder_3    1. Sets the name of the strategy  2. Sets the description of the strategy |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheGeneralScreen)

tog_minus        [Understanding the Default properties screen](javascript:HMToggle('toggle','UnderstandingTheDefaultPropertiesScreen','UnderstandingTheDefaultPropertiesScreen_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Default properties screen Layout**  The Default properties screen is where you can set the default values for your custom strategy properties.    Strategy_Builder_4    1. Per default only the **Calculate** section is visible, click the **More properties** to expand the selection to include all strategy default properties as well to set for your Builder script.     |  |  | | --- | --- | | Calculate | Sets the **Calculation Mode** for the strategy. Possible values are "On Each Tick," "On Price Change," or "On Bar Close" | | Entries per direction | Sets the maximum number of entries allowed per direction while a position is active based on the "Entry handling" property | | Entry handling | Sets the manner in which entry orders are handled. If set to "AllEntries", the strategy will process all entry orders until the maximum allowable entries set by the "Entries per direction" property have been reached while in an open position. If set to "UniqueEntries", the strategy will process entry orders until the maximum allowable entries set by the "Entries per direction" property per each uniquely named entry have been reached. | | Exit on close | When enabled, open positions will be closed on the last bar of a session | | Exit on session close seconds | Sets the number of seconds prior to the end of a session at to close any open positions held by the strategy | | Fill Limit Orders on Touch | Enables the filling of limit orders when touched for the historical portion of the chart | | Maximum Bars Look Back | Sets the maximum number of historical bars to use for strategy calculations. The TwoHundredFiftySix setting is the most memory friendly | | Minimum Bars Required | Sets the minimum number of historical bars required to start taking trades | | Order Fill Resolution | Sets the way that simulated historical orders will be processed by the strategy. See the [Understanding Historical Fill Processing](https://ninjatrader.com/support/helpGuides/nt8/understanding_historical_fill_.htm) page for more information. | | Real-time error handling | Defines the behavior of a strategy when a strategy generated order is returning in a "Rejected" state. See the [Real-time Error Handling](https://ninjatrader.com/support/helpGuides/nt8/realtimeerrorhandling.htm) page for more information. | | Slippage | Sets the slippage amount in ticks for the historical portion of the chart | | Start Behavior | Sets the starting behavior of the strategy, based upon the account position. See the [Syncing Account Positions](https://ninjatrader.com/support/helpGuides/nt8/syncing_account_positions.htm) page for more information. | | Stops and Targets | Sets how stop and target orders are submitted | | Time in force | Sets the order's time in force. Possible values are DAY and GTC | | Trace orders | Enables sending more detailed order debug info to the NinjaScript [output window](https://ninjatrader.com/support/helpGuides/nt8/output.htm) | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheDefaultPropertiesScreen)

tog_minus        [Understanding the Additional data screen](javascript:HMToggle('toggle','UnderstandingTheAdditrionalDataScreen','UnderstandingTheAdditrionalDataScreen_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Additional data screen Layout**  The Additional data screen is where you can optionally select additional instrument data or custom series for your strategy.     |  | | --- | | **Critical**: You will want to make sure to add any additional series in this section that a hosted / called MultiSeries indicator in your Builder script would use, such as for example the [Pivots](https://ninjatrader.com/support/helpGuides/nt8/pivots.htm), [Camarilla Pivots](https://ninjatrader.com/support/helpGuides/nt8/camarilla_pivots.htm) or [Fibonacci Pivots](https://ninjatrader.com/support/helpGuides/nt8/fibonacci_pivots.htm) indicators. |     Strategy_Builder_5    1. Press the **add** button to be able to configure a new series to add  2. Press the **edit** button to be able to configure an existing series  3. Press the **remove** button to be able to remove an existing series    **Data Series Selector Layout**  Select your instrument data series to add here    Strategy_Builder_6     |  |  | | --- | --- | | Use primary instrument | Checking this will use the primary instrument the strategy is applied to | | Instrument | Select your instrument from the favorite or list selector or by using the search feature (press the magnifying glass) | | Price based on | Selects the price type the data series is based on, possible values are Last, Bid, Ask | | Type | Selects the bars type your series will use, possible values for the Builder interface are -    •Tick  •Minute  •Day  •Week  •Month  •Year  •Volume  •Range  •Second | | Value | Sets the bars period type value for your series |     **Custom Series Selector Layout**  Select your custom series to add here    Strategy_Builder_7   |  |  | | --- | --- | | Name | Set the name for your custom series | | Type | Selects the data type of the custom series, possible values for the Builder interface are -    •Bool  •Double  •DateTime  •Int  •String | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheAdditrionalDataScreen)

tog_minus        [Understanding the Inputs and Variables screen](javascript:HMToggle('toggle','UnderstandingTheInputsAndVariablesScreen','UnderstandingTheInputsAndVariablesScreen_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Inputs and Variables screen Layout**  The Inputs and Variables screen allows you to define the user inputs of your strategy. User inputs are important if you require input values that may vary the performance of your strategy. If for example you have a simple moving average cross over system, you may want to create an input for the fast moving average and another for the slow moving average. This then allows you to change the values of the moving averages at run time from the UI. Inputs are also required if you plan to use the NinjaTrader [Strategy Analyzer's optimization](https://ninjatrader.com/support/helpGuides/nt8/strategy_analyzer.htm) capabilities.    Strategy_Builder_8    1. Press the **add** button to add a new user input.  2. Press the **edit** button to edit an existing, selected user input.  3. Press the **remove** button to remove the selected user input.     |  |  | | --- | --- | | Name | Set the name for your user input | | Type | Selects the data type of the user input, possible values for the Builder interface are -    •Bool  •Double  •String  •Int  •Time | | Default | Set the default value your user input will have | | Min | Set the minimum value your user input will have | | Description | Enter an optional description for your user input here |     Strategy_Builder_9    1. Press the **add** button to add a new user variable.  2. Press the **edit** button to edit an existing, selected user variable.  3. Press the **remove** button to remove the selected user variable.     |  |  | | --- | --- | | Name | Set the name for your user variable | | Type | Selects the data type of the user variable, possible values for the Builder interface are -    •Bool  •Double  •String  •Int  •Time | | Default | Set the default value your user variable will have |      |  | | --- | | **Note**: If an input is named the same as the generated code for an indicator, the strategy will not be able to successfully compile. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheInputsAndVariablesScreen)

tog_minus        [Understanding the Conditions and Actions screen](javascript:HMToggle('toggle','UnderstandingTheConditionsAndActionsScreen','UnderstandingTheConditionsAndActionsScreen_ICON'))

|  |
| --- |
| **Conditions and Actions screen Layout**  The Conditions and Actions screen allows you to set conditions and subsequent actions that control the flow of your strategy.    *Conditions* - Take the specified action when true  *Actions* - Execute an action (submit orders, draw objects on the chart etc ...) based on its parent condition evaluating to true    Via the Builder, you can have an unlimited set of conditions with related actions and you also group conditions into a condition group (for example for a certain set of filter rules like time)    Conditions and condition groups are created using the [Condition Builder](https://ninjatrader.com/support/helpGuides/nt8/strategybuilder_condition_builder.htm). Actions are specified by the [Strategy Actions](https://ninjatrader.com/support/helpGuides/nt8/actions.htm) window.    Strategy_Builder_10    1. Selects **if all** of the individual conditions have to be met in order to trigger an action, or **if any** will be sufficient.  2. Displays the conditions associated with the currently selected condition set  3. Adds, opens condition grouping(\*), edits or removes a condition (a double click on selected item will also allow editing)  4. Displays the actions associated with the currently selected condition set  5. Adds, edits or removes an action (a double click on selected item will also allow editing)  6. Selects the condition set you wish to edit    \* For an example on working condition groups, please see "How to create a Time Filter' in the [Condition Builder](https://ninjatrader.com/support/helpGuides/nt8/strategybuilder_condition_builder.htm) section    You can copy and paste conditions from one set to another and you can even save a condition set as a template and load for future use via the right mouse button click context menu as show in the image below. To save a condition set as a template, select the **Save As...** menu item and then to re-use it in another strategy or condition set at a later time, select the**Load...** menu item.    Strategy_Builder_11 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheConditionsAndActionsScreen)

tog_minus        [Understanding the Stops and Targets screen](javascript:HMToggle('toggle','UnderstandingTheStopsAndTargetsScreen','UnderstandingTheStopsAndTargetsScreen_ICON'))

|  |
| --- |
| **Stops and Targets screen Layout**  The Stops and Targets allows you to set stop loss, trail stop, parabolic stop (R15 and higher) and profit target orders that are automatically submitted and managed once your strategy opens a position.    Strategy_Builder_12    1. Displays stops and targets associated with your strategy  2. Adds a stop or target to your strategy  3. Edits the selected stop or target in your strategy (a double click on the selected item will also allow editing)  4. Removes the selected stop or target from your strategy |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheStopsAndTargetsScreen)

tog_minus        [Understanding the Finish screen](javascript:HMToggle('toggle','UnderstandingTheFinishScreen','UnderstandingTheFinishScreen_ICON'))

|  |
| --- |
| **Finish screen Layout**  Once you reach this screen you are finished with developing your strategy. Press the **Finish** button to compile your strategy which will then be ready for [backtesting](https://ninjatrader.com/support/helpGuides/nt8/backtest_a_strategy.htm) or [live execution](https://ninjatrader.com/support/helpGuides/nt8/running_ninjascript_strategies.htm).    Strategy_Builder_13 |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?builder_screens.htm#UnderstandingTheFinishScreen)

|  |  |
| --- | --- |
| **Navigation:**  [Operations](https://ninjatrader.com/support/helpGuides/nt8/operations.htm) > [Charts](https://ninjatrader.com/support/helpGuides/nt8/charts.htm) > [Order Flow +](https://ninjatrader.com/support/helpGuides/nt8/order_flow_plus.htm) >  **Order Flow Volume Profile** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/order_flow_vwap.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/order_flow_plus.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/order_flow_trade_detector.htm) |

**Description**  
Order Flow Volume profile tools are available as both an Indicator and Drawing Tool and along with the suite of the ['Order Flow +' tools](https://ninjatrader.com/support/helpGuides/nt8/order_flow_plus.htm) only available to NinjaTrader lifetime license holders. The indicator is used to plot singular static profiles containing a certain defined range of data or repeating profiles on a per bar or per session basis. The drawing tool allows you to easily define both the start and end point to create a custom profile for any bar range on a chart.

tog_minus        [Order Flow Volume Profile Concepts](javascript:HMToggle('toggle','OrderFlowVolumeProfileConcepts','OrderFlowVolumeProfileConcepts_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Profile Types**  Order Flow Volume Profile has many settings which can be customized to achieve several types of profiles.  There are 3 modes for selecting what you want to use to generate the profile.     |  |  | | --- | --- | | **Volume Profile**    Volume profiles display last traded volume data using your selected source data 'resolution' is used to build the profile. | vp2 | | **Price Profile**    Price profiles displays a letter or block if underlying trades inside of a specific time window. These time windows are 30 minute in length. If a single tick occurs inside that time window at a price a letter or block will be plotted. In letter display mode the 'A' letter starts at 8:00 CST. | vp3 | | **Tick Profile**  Tick profiles display the count of 'ticks' or trades that occur at a specific price. This produces a quantity agnostic profile which display pure trading activity irregardless of volume. | vp4 |     **Profile Components**  vp6    1.**Profile**: Overlaid behind the bars each bar represents the profile per price. Multiple price levels can be aggregated using 'Ticks per level'.  2.**POC**: The point of control (POC) is the single largest data point in the profile.  3.**Value area**:  The range where 68% (configurable) of the volume traded, can be referenced as 'VA' and has opacity settings to visually separate this area from the rest of the profile.  4.**Range**: The highest price and lowest price of the profile.  5.**Profile** **Summary**: The total volume and range of a volume profile. Range metric is configurable.    **Profile Periods**  There are 3 methods to select the 'period' you which to generate a profile for, each one renders slightly differently. It is not uncommon to use multiple profile types on the same chart as each profile gives you a different view on the underlying data.     |  |  |  | | --- | --- | --- | | **Session** | **Bars** | **Composite** | | vp7 | vp8 | vp9 | | This is a repeating profile created for each session defined. The profile by default is displayed below the bars and across all bars that make up the session. You can group multiple sessions by defining the property 'Sessions' greater then 1 (count is run from left to right, so forward). | When Bars is selected, one profile per bar is rendered. Space is added between the bars so that the profile is readable. This space is defined in the 'Visual' section below 'Profile width (px)' which is a defaulted to 100. You can group multiple bars by defining the property 'Bars' greater then 1. | A composite profile is a single profile drawn as an overlay on the entire chart that is composed of data defined by the 'Compose by' sub property that is exposed when this mode is selected. |     **Display Modes**  Multiple display modes can be used. By default there are 5 render modes available which is changed with the 'Display mode' property on the indicator.     |  |  | | --- | --- | | **Standard**    Single defined color profile rendering display volume by price information. The color can be selected by modifying the 'Color for profile' property. There are two properties to define the opacity. 'Profile opacity' sets the opacity above and below the value area and 'Value area opacity' sets the opacity used when drawing the area in the value area. | vp10 | | **Buy sell**    The volume bar now split by buy volume and sell volume as classified per the Delta type property.    Note: This mode requires 'Tick' resolution data. | vp11 | | **Delta**    Displays the standard volume profile and overlays buy/sell data delta as classified per the Delta type property. This display mode highlights imbalances in buy/sell volume.    Note: This mode requires 'Tick' resolution data | .vp12 | | **Heat**    Displays a gradient in the profile based on the highest volume and the lowest volume in the profile. High volume areas are emphasized in this mode. | vp13 | | **Outline**    Displays the volume profile with only the outline of the profile being plotted. | vp14 | | **Time color**    Every 30 minutes a new color is plotted and displayed in the profile. The profile color is defined in the properties 'Times' section. You can see what time of day the majority of the volume occurred in the mode. | vp15 |     **Initial Balance**  The initial balance is an optional feature which will enable plots to show you what the current range, POC, and Value area a set number of minutes into the session. A typical use case is so that you can compare what the POC and VA was early on in the profile and make a comparison to where it ended up. The initial balance range needs to have the property 'Initial balance time (in minutes)' set to a non-zero value. Once 'Initial balance time' is defined the there are 3 lines which will be enabled and can be configured, you can enable only the lines you wish to see to display the 'Initial Range', 'Initial POC', and 'Initial Value Area'.    vp16    1.Initial Range: This is the range of the profile used in the initial profile calculation that was traded during the 'initial balance time (in minutes)'  2.Initial POC: The value of where the point of control (POC) is located during the 'initial balance time (in minutes)'  3.Initial Value Area: The value area during the 'initial balance time (in minutes)'    **Developing POC and Value Area**  The developing POC and Value Area is an optional feature which when enabled will show you bar by bar how these value change overtime for the profile.    vp17    1.Developing POC: For each bar in the chart the value of the POC is plotted so you can see the change over time.  2.Developing Value Area:For each bar in the chart the value of the value area high and value area low so you can see the change over time.    **Extended "Naked" Point of Control and Value Area**  The extended "Naked" Point of Control and Value Area will take any POC of Value Area of a profile and plot it forward from the end time of the profile until such time as the price is traded again in the future. Uncovered or 'naked' are expected to be 'filled' sometime in the future and may be an area of support or resistance. There visibility and plot properties are set in the 'Lines' section    vp18    1.Naked POC  2.Naked Value Area     |  | | --- | | **Note**: If a line is enabled, but it is not seen, it may be under another enabled line. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order_flow_volume_profile.htm#OrderFlowVolumeProfileConcepts)

tog_minus        [Understanding Price Profiles](javascript:HMToggle('toggle','UnderstandingPriceProfile','UnderstandingPriceProfile_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Price Profiles**  Price profiles displays a letter or block if the underlying instrument trades inside of a specific time window. These time windows are 30 minute in length and if a single trade occurs in the time range a single block or letter is plotted. NinjaTrader supports price charts on intraday charts only. Its recommended to use a minute base chart such as 60 minute, 240 minute, 480 minute base chart types. Typically the 'Hide Bars' property is enabled for price profiles. This allows you to focus on the price profile on its own, the candles are no longer needed once you become proficient reading the price profile since both time and price is displayed once you memorize the time/color chart below.    To easily see where the session has it's Open / (developing) Close NinjaTrader displays separate O / C markers. These letters are then excluded in the time/color letter progression for the session.    **Scaling the Price Profile**  Profile scaling may be required to get your price profile to render as you expect. When rendering in letter mode if there is not enough space to render a letter which would be large enough to read on the screen the profile will fall back to block rendering mode. In this case we need to adjust chart scaling and ticks per level depending on the instrument you wish to plot the price profile on.    1.To adjust the distance between profiles, use the time scale drag in the time axis to compress and decompress the underling bars. If you've reached the maximum your able to adjust using the time axis, try adjusting the underling bars up or down as needed, start with a 60 minute chart and increase the minute range by double each time to compress the chart in the time axis.  2.To adjust the price scale use left click and drag in the price label area to adjust price scaling and put the chart into 'fixed' scaling mode. You can hold down left mouse and drag and CTRL on the keyboard to move the range being displayed to focus in on the area you wish to focus on. See the section ['Navigating a Chart'](https://ninjatrader.com/support/helpGuides/nt8/navigating_a_chart.htm) for more information on working with price scales.    **Price profile times**  It is important to look into the 'Order Flow + Volume Profile' properties, specifically the 'Times' section to understand the time slices reported by the tool and what color and letter is associated. The time displayed in converted to your local time zone and is a fixed frame of referenced from starting at 8:00 CST. This time is arbitrarily picked and provides all instruments to share the same time slice, therefore the letter 'A' and associated color will always be the same calendar day and not effected by the individual instrument, this allows comparison between charts based on a fixed time scale. The color for each time slice is customizable, once you have it defined     |  | | --- | | **Note**: Only intraday charts are supported. Daily, weekly, or monthly charts are not supported at this time. |     **Reference Table**   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | A | 8:00 - 8:30 AM CST | | B | 8:30 - 9:00 AM CST | | D | 9:00 - 9:30  AM CST | | E | 9:30 - 10:00 AM  CST | | F | 10:00 - 10:30 AM CST | | G | 10:30 - 11:00 AM CST | | H | 11:00 - 11:30 AM CST | | I | 11:30 - 12:00 PM CST | | J | 12:00 - 12:30 PM CST | | K | 12:30 - 1:00 PM CST | | L | 1:00 - 1:30 PM CST | | M | 1:30 - 2:00 PM CST | | N | 2:00 - 2:30 PM CST | | P | 2:30 - 3:00 PM CST | | Q | 3:00 - 3:30 PM CST | | R | 3:30 - 4:00 PM CST | | S | 4:00 - 4:30 PM CST | | T | 4:30 - 5:00 PM CST | | U | 5:00 - 5:30 PM CST | | V | 5:30 - 6:00 PM CST | | W | 6:00 - 6:30 PM CST | | X | 6:30 - 7:00 PM CST | | Y | 7:00 - 7:30 PM CST | | Z | 7:30 - 8:00 PM CST | | |  |  | | --- | --- | | a | 8:00 - 8:30 PM CST | | b | 8:30 - 9:00 PM CST | | d | 9:00 - 9:30  PM CST | | e | 9:30 - 10:00 PM  CST | | f | 10:00 - 10:30 PM CST | | g | 10:30 - 11:00 PM CST | | h | 11:00 - 11:30 PM CST | | i | 11:30 - 12:00 PM CST | | j | 12:00 - 12:30 AM CST | | k | 12:30 - 1:00 AM CST | | l | 1:00 - 1:30 AM CST | | m | 1:30 - 2:00 AM CST | | n | 2:00 - 2:30 AM CST | | p | 2:30 - 3:00 AM CST | | q | 3:00 - 3:30 AM CST | | r | 3:30 - 4:00 AM CST | | s | 4:00 - 4:30 AM CST | | t | 4:30 - 5:00 AM CST | | u | 5:00 - 5:30 AM CST | | v | 5:30 - 6:00 AM CST | | w | 6:00 - 6:30 AM CST | | x | 6:30 - 7:00 AM CST | | y | 7:00 - 7:30 AM CST | | z | 7:30 - 8:00 AM CST | | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order_flow_volume_profile.htm#UnderstandingPriceProfile)

tog_minus        [Understanding Composite Profiles](javascript:HMToggle('toggle','Composite','Composite_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Composite Profiles** A composite profile is a singular large profile rendered on either the left or right side of the chart containing data as defined by the composite properties which become visible once selected. Order Flow Volume Profile has many settings which can be customized to achieve several types of profiles. Its not uncommon to apply a composite profile and at the same time adjust the chart margin to allow for spacing on the right side of the chart. To do this right click on the chart and select 'Properties', adjust the 'Right side margin' property to a higher value. Depending on the 'Profile width (%)' selection of the chart will determine the number of pixels you will need to enter, work in 100 px increments until you find the right size for the configured chart.    You have several options to determine what data to be included in the composite profile:     |  |  | | --- | --- | | Weeks Back | The number of weeks back to be factored in this profile. | | Days Back | The number of days back from the current day to include in the profile. | | Start Date | Manually specify a start date for the profile. | | Chart | All the data available in the charts 'Start date' and 'End Date' are displayed in the profile. | | Visible Screen Range | The profile will dynamically update based on the data determined from the first bar in view in the chart and the last bar in view on the chart. Moving the chart from left to right will change what data is displayed in the profile. |     vp23     |  | | --- | | **Note**: Composite profiles using 'minute' data being placed on a chart which has bar ranges less then 1 minute is not compatible. In this scenario tick resolution must be used. |      |  | | --- | | **Note**: You want to make sure that the data you wish to compose a profile of actually exists on your chart to have a complete profile. You may need to right click on the chart > [Data Series > and increase the days to load](https://ninjatrader.com/support/helpGuides/nt8/working_with_price_data.htm) for the chart. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order_flow_volume_profile.htm#Composite)

tog_minus        [Understanding the Order Flow Volume Profile Drawing Tool](javascript:HMToggle('toggle','DrawingTool','DrawingTool_ICON'))

|  |  |
| --- | --- |
| **Order Flow Volume Profile Drawing Tool** The Order Flow + Volume Profile Drawing Tool allows you to configure a profile using all the similar settings and configurations which are applicable from the indicator and manually draw out and specify a start and end time for the profile right on the chart. Settings which do not apply are not available on the drawing tool.    **Drawing a Profile**  To draw a profile bounds, in the Drawing tool menu select 'Order Flow Volume Profile' from the menu and single click to define the start point of the profile. An outline box will begin to be displayed where you drag your mouse to the ending point for the profile bounds and left click again to complete the draw operation. The bounds box will stay centered and overlaid of the data which will be display the profile. Once the bounds have been selected NinjaTrader will attempt to load the requested data ('Calculating...' text will appear inside the outline box), by default minute resolution data is used however you can change the properties to use 'Tick' resolution data depending on the type of charts you will be drawing on. Minute works well for daily or any larger time frame chart whereas on more granular intraday charts you will want to change the default to 'Tick' resolution. The section below will detail how to change the default so that future drawing do not require changing the property each time.    vp19    **Configuring the Drawing Tool**  Double clicking the drawing tool or single left click + right click and 'Properties' will allow you to customize the drawing tools various properties. As there are several properties you will want to set a preset for this drawing tool as you have it setup as you prefer. To do this in the 'Drawing Objects' select 'template' and then save, if you save a template with the name 'Default' this will be the template used moving forward on any newly created volume profile. More detailed info on NinjaTrader's drawing tools and operating them, could be reviewed [here](https://ninjatrader.com/support/helpGuides/nt8/working_with_drawing_tools__ob.htm).    vp20     |  | | --- | | **Note**: The Order Flow Plus Volume Profile drawing tool will not work on time ranges (historical data) outside the current selected playback range. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order_flow_volume_profile.htm#DrawingTool)

tog_minus        [Order Flow Volume Profile 'Set up' Parameters](javascript:HMToggle('toggle','OrderFlowVWAPParameters','OrderFlowVWAPParameters_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Profile type | This is the top level parameter for selecting the type of profile you would like to generate. Each one of these profiles can be completely different since its constructed with different source data. Volume and tick share rendering options however price is unique in that its rendered using blocks or letters.     |  |  | | --- | --- | | Volume | Volume profiles display last traded volume data using your selected source data 'resolution' is used to build the profile. | | Tick | Tick profiles display the count of 'ticks' or trades that occur at a specific price. This produces a quantity agnostic profile which display pure trading activity irregardless of volume. | | Price | Price profiles displays a letter or block if underlying trades inside of a specific time window. These time windows are 30 minute in length. If a single tick occurs inside that time window at a price a letter or block will be plotted. In letter display mode the 'A' letter starts at 8:00 CST. | | | Display mode | Defines how the profile will be rendered and what data is impressed.    Volume & Tick Display Modes:     |  |  | | --- | --- | | Standard | Single defined color profile rendering display volume by price information. | | Buy sell | The volume bar now split by buy volume and sell volume as classified per the Delta type property.    Note: This mode requires 'Tick' resolution data. | | Delta | Displays the standard volume profile and overlays buy/sell data delta as classified per the Delta type property. This display mode highlights imbalances in buy/sell volume.    Note: This mode requires 'Tick' resolution data. | | Heat | Displays a gradient in the profile based on the highest volume and the lowest volume in the profile. High volume areas are emphasized in this mode. | | Outline | Displays the volume profile with only the outline of the profile being plotted. | | Time color | Every 30 minutes a new color is plotted and displayed in the profile. The profile color is defined in the properties 'Times' section. |     Price Display Modes:     |  |  | | --- | --- | | Letters | A letter is displayed if the last price traded anywhere in the price range in a 30 minute window. The letter definitions can be found in the 'Times' category in the indicator properties. Capital 'A' letter starts at 8:00 AM CST. When not enough space is available for a 'letter' to be rendered, the profile we default to 'Boxes' display mode below. | | Boxes | Instead of displaying a letter as per 'letter' mode above, only the corresponding color to the letter is displayed. This modes is used when you want to display more data on the screen. | | | Delta type (Only visible when Buy Sell or Delta Display mode is selected) | Sets how the delta is calculated for buy / sell aggressor classification.     |  |  | | --- | --- | | BidAsk | Last trade at the ask or higher is considered buying volume, Last trade at the bid or lower selling volume. | | UpDownTick | Last trade happens while Ask > Last Ask is considered buying, Last trade happens while Bid < Last Bid considered selling, all volume in between is added to the prior direction - this mode is an important proxy for markets / data providers where best bid / ask information is not available with last price tick data | | | Profile period | |  |  | | --- | --- | | Sessions | Each session as defined by the Charts Data Series or the manual Trading hours selected gets a rendered Volume Profile. The profile by default is displayed below the bars and across all bars that make up the session. You can group multiple sessions by defining the property 'Sessions' greater then 1 (count is run from left to right, so forward). | | Bars | When Bars is selected, one profile per bar is rendered. Space is added between the bars so that the profile is readable. This space is defined in the 'Visual' section below 'Profile width (px)' which is a defaulted to 100. You can group multiple bars by defining the property 'Bars' greater then 1. | | Composite | A composite profile is a single profile drawn as an overlay on the entire chart that is composed of data defined by the 'Compose by' sub property that is exposed when this mode is selected.    Compose by:     |  |  | | --- | --- | | Weeks Back | The number of weeks back to be factored in this profile. | | Days Back | The number of days back from the current day to include in the profile. | | Start Date | Manually specify a start date for the profile. | | Chart | All the data available in the charts 'Start date' and 'End Date' are displayed in the profile. | | Visible Screen Range | The profile will dynamically update based on the data determined from the first bar in view in the chart and the last bar in view on the chart. Moving the chart from left to right will change what data is displayed in the profile. |      |  | | --- | | **Note**: You want to make sure that the data you wish to compose a profile of actually exists on your chart to have a complete profile. You may need to right click on the chart > Data Series > and increase the days to load for the chart. | | | | Trading hours | Only data defined in the trading hour template will be used to calculate the volume profile. The default will use all the data as displayed in the chart.     |  | | --- | | **Note**: This setting can only be used as a filter, meaning that all the base data must exist in the chart. E.G. A chart set using an ETH template and using a RTH template volume profile will work correctly, the opposite would not as the indicator will not load the additional data needed for outside RTH chart hours in our example. | | | Resolution | |  |  | | --- | --- | | Minute | The profile is generated from 1 minute data for the profile period selected. This data is much faster to load and works as a sensible default to generate and approximate profile. Volume data is evenly divided amongst each price between the high and the low of the 1 minute bar. | | Tick | The profile is generated using 1 tick data. This will be the most accurate and most resource intensive. Only select tick when you know the range of tick data you will be processing is limited. | | | Ticks per level | Sets the level of aggregation for individual price levels, i.e. if price levels should be merged together, default 1 – so each price level is seen and accounted for individually in the profiles | | Value area (%) | The default of 68 percent for Value Area will display a 1 standard deviation range from the point of control and is customizable by the user. Value area is shown using a different opacity in the profile and additionally has a 'line' and 'label' which is drawn at the Value area high and Value area low. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order_flow_volume_profile.htm#OrderFlowVWAPParameters)

tog_minus        [Order Flow Volume Profile 'Visual' Parameters](javascript:HMToggle('toggle','OFVPVisual','OFVPVisual_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Profile alignment | |  |  | | --- | --- | | Left | The profile is drawn using the left edge. | | Right | The profile is drawn using the right edge. | | | Display in margin | Only available for Profile period Composite, if selected will display the profile and including lines in the right side margin | | Profile width (%) or (px) | This will be in pixels or in percentage, depending on the profile period selection. With percentage it indicates what percentage of space between sessions the bars will take. With pixels it will be how many pixels of space the bars will take. | | Profile opacity | This is the base opacity used in to render the profile. | | Show POC | This will shade the POC (Point of Control) the color you have selected in the 'Color for POC' property. | | Show Value area | Enables the display of the value area. | | Value area opacity | Volume bars which are within the value area are plotted with less opacity then the profile, allowing them to standout. To disable this, set the opacity to the same value as the 'Profile opacity' | | Color for POC | The color which is used to define the highest volume (Point of Control) seen for the profile. | | Color for profile | In 'Standard' draw mode, this property defines the color for the profile. | | Color for buy | In 'BuySell' and 'Delta' draw mode, this property defines the color for buy classified orders. | | Color for sell | In 'BuySell' and 'Delta' draw mode, this property defines the color for the sell classified orders. | | Color for high heat | In 'Heat' draw mode, this property defines the top gradient for the highest volume seen in the profile. | | Color for low heat | In 'Heat' draw mode, this property defines the bottom gradient for the lowest volume seen in the profile. | | Hide bars | Convenience method to hide the underlying bars enhancing focus on the profiles. This can be done using this option via the indicator or by manually setting the DataSeries bar color to 'Transparent'. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order_flow_volume_profile.htm#OFVPVisual)

tog_minus        [Order Flow Volume Profile 'Lines' Parameters](javascript:HMToggle('toggle','OFVP_Lines','OFVP_Lines_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The 'Lines' section defined all available and configurable line plots. Each line property once extended can have its Color, Dash style, Width, Visible, and Label property defined.       |  |  | | --- | --- | | POC | Enable and customize display of the Point of Control line. Please see 'Profile Components' section in the Order Flow Concepts section for more information. | | Value area (%) | Enable and customize display of the Value area lines. Please see 'Profile Components' section in the Order Flow Concepts section for more information. | | Range | Enable and customize display of the Range lines. Please see 'Profile Components' section in the Order Flow Concepts section for more information. | | Initial POC | Enable and customize display of the Initial Point of Control line. Please see 'Initial Balance' section in the Order Flow Concepts section for more information. | | Initial Value Area | Enable and customize display of the Initial Value Area lines. Please see 'Initial Balance' section in the Order Flow Concepts section for more information. | | Initial Balance Range | Enable and customize display of the Initial Balance range lines. Please see 'Initial Balance' section in the Order Flow Concepts section for more information. | | Developing POC | Enable and customize display of the Developing POC lines. Please see 'Developing POC and Value Area'  section in the Order Flow Concepts section for more information. | | Developing Value Area | Enable and customize display of the Developing Value Area lines. Please see 'Developing POC and Value Area'  section in the Order Flow Concepts section for more information. | | Extended naked POC | Enable and customize display of the Extended POC line. Please see 'Extended 'Naked' POC and Value Area'  section in the Order Flow Concepts section for more information. | | Extended naked Value Area | Enable and customize display of the Extended Value Area lines. Please see 'Extended 'Naked' POC and Value Area'  section in the Order Flow Concepts section for more information. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order_flow_volume_profile.htm#OFVP_Lines)

tog_minus        [Order Flow Volume Profile 'Label' Parameters](javascript:HMToggle('toggle','OFVPLabel','OFVPLabel_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The 'Lines' section defined all available and configurable line plots. Each line property once extended can have its Color, Dash style, Width, Visible, and Label property defined.       |  |  | | --- | --- | | Font | Font settings used for labels and summary data. | | Show volume labels | Optionally display the underlying data that makes up the volume profile. To visually see text data you may be required to increase 'Ticks per level' to allow more space for the text to render depending on the scale of the chart. | | Show profile summary | This shows total volume, and range information as summary statistics in the bottom left of the profile range. | | Summary display unit | Configure the range display unit for summary data (Price, Percent, Ticks, Currency, Pips) | | Color for volume labels | Sets the text color used when 'Show volume labels' is enabled. | | Color for profile summary | Sets the text color used when 'Show profile summary' is enabled. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order_flow_volume_profile.htm#OFVPLabel)

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/support/helpGuides/nt8/common.htm) > [Charts](https://ninjatrader.com/support/helpGuides/nt8/chart.htm) > [Rendering](https://ninjatrader.com/support/helpGuides/nt8/rendering.htm) >  **OnRender()** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/oncalculateminmax.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/rendering.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/onrendertargetchanged.htm) |

**Definition**

Used to render custom drawing to a chart from various chart objects, such as an [Indicator](https://ninjatrader.com/support/helpGuides/nt8/indicator.htm), [DrawingTool](https://ninjatrader.com/support/helpGuides/nt8/drawingtool.htm) or [Strategy](https://ninjatrader.com/support/helpGuides/nt8/strategy.htm).

|  |
| --- |
| **Notes**:  1.This method uses the 3rd party SharpDX library to render custom Direct2D Text and Shapes.  For a walk through for using the **SharpDX**, please see the educational resource [Using SharpDX for Custom Chart Rendering](https://ninjatrader.com/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm)  2.The **OnRender()**method frequently runs once the [State](https://ninjatrader.com/support/helpGuides/nt8/state.htm) has reached **State.Realtime** in response to market data updates or a user interacting with the chart (e.g., clicking, resizing, rescaling, etc.)  3.For performance optimizations, the timing of the calls to **OnRender()** are buffered to at least 250ms, and re-renders once internal logic determines that values may be out-of-date.  See also [ForceRefresh()](https://ninjatrader.com/support/helpGuides/nt8/forcerefresh.htm) for more details  4.When using the [Strategy Analyzer](https://ninjatrader.com/support/helpGuides/nt8/strategy_analyzer.htm), **OnRender()** does **NOT** call until you switch to the "Chart" display and renders from **State.Terminated**.  As a result, this method should **NOT** be relied on for historical Strategy backtesting logic and should **ONLY** be used for rendering purposes  5.Unlike market data events and strategy order related events, there is **NO** guarantee that the *barsAgo* indexer used for [Series<T>](https://ninjatrader.com/support/helpGuides/nt8/seriest.htm) objects are in sync with the current bars in progress.  As a result, you should favor using an absolute index method to look up values (e.g.,[<series>.GetValueAt()](https://ninjatrader.com/support/helpGuides/nt8/getvalueat.htm), [Bars.GetOpen()](https://ninjatrader.com/support/helpGuides/nt8/getopen.htm), etc)  6.While **OnRender()** is an excellent means for customizing and enhancing indicators and strategies, its application can easily be abused, resulting in unforeseen performance issues which you may not catch until the right conditions (e.g., in the hands of your users during an FOMC event)  7.Please limit any calculations or algorithms you may be tempted run in OnRender() simply to rendering. You should always favor precomputed values and store them for rendering later as the preferred approach to working with the OnRender() method (e.g., reusing brushes, passing values from [OnBarUpdate()](https://ninjatrader.com/support/helpGuides/nt8/onbarupdate.htm), etc.).  See also [OnRenderTargetChanged()](https://ninjatrader.com/support/helpGuides/nt8/onrendertargetchanged.htm) method for more information on reusing Brushes  8.If you are using this method as an opportunity to "hook" onto a user related event, such as when a user selects a 3rd party control, you should alternatively consider using the events of that control independent of official NinjaScript events. See also [TriggerCustomEvent()](https://ninjatrader.com/support/helpGuides/nt8/triggercustomevent.htm) |

**Method Return Value**

This method does not return a value

**Syntax**

protected override void OnRender(ChartControl chartControl, ChartScale chartScale)  
{  
   
}

|  |
| --- |
| **Warning**:  Each DirectX [render target](https://ninjatrader.com/support/helpGuides/nt8/rendertarget.htm) requires its own brushes. You must create a brushes directly in **OnRender()** or using [OnRenderTargetChanged()](https://ninjatrader.com/support/helpGuides/nt8/onrendertargetchanged.htm).  If you do not you will receive an error at run time similar to:   ***"A direct X error has occured while rendering the chart: HRESULT: [0x88990015], Module: [SharpDX.Direct2D1], ApiCode: [D2DERR\_WRONG\_RESOURCE\_DOMAIN/WrongResourceDomain], Message: The resource was realized on the wrong render target. : Each DirectX render target requires its own brushes. You must create brushes directly in OnRender() or using OnRenderTargetChanged().***    Please see [OnRenderTargetChanged()](https://ninjatrader.com/support/helpGuides/nt8/onrendertargetchanged.htm) for examples of a brush that needs to be recalculated, or the example below of recreating a static brush. |

**Method Parameters**

|  |  |
| --- | --- |
| chartControl | A [ChartControl](https://ninjatrader.com/support/helpGuides/nt8/chartcontrol.htm) object (the chart's bar-related properties and x-axis) |
| chartScale | A [ChartScale](https://ninjatrader.com/support/helpGuides/nt8/chartscale.htm) object (the chart's y-axis) |

|  |
| --- |
| **Tips**:  •Please see the help guide topic on [Working with Brushes](https://ninjatrader.com/support/helpGuides/nt8/working_with_brushes.htm) for general information on using brushes and advanced brush concepts  •If you are using standard [Plots](https://ninjatrader.com/support/helpGuides/nt8/plots.htm) along with custom rendering from an indicator or strategy, you will need to ensure to call the **base.OnRender()** method for those plots to display. |

**Examples**

| ns **Using a static SharpDX Brush to render a rectangle on the chart panel** |
| --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // implicitly recreate and dispose of brush on each render pass   using (SharpDX.Direct2D1.SolidColorBrush dxBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.Blue))   {     RenderTarget.FillRectangle(new SharpDX.RectangleF(ChartPanel.X, ChartPanel.Y, ChartPanel.W, ChartPanel.H), dxBrush);   } } |

| ns **Calling the base.OnRender() method to ensure Plots are rendered along with custom render logic** |
| --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // call the base.OnRender() to ensure standard Plots work as designed   base.OnRender(chartControl, chartScale);     // custom render logic } |

| ns **Using multiple SharpDX objects to override the default plot appearance** | |
| --- | --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // get the starting and ending bars from what is rendered on the chart   float startX = chartControl.GetXByBarIndex(ChartBars, ChartBars.FromIndex);   float endX = chartControl.GetXByBarIndex(ChartBars, ChartBars.ToIndex);     // Loop through each Plot Values on the chart   for (int seriesCount = 0; seriesCount < Values.Length; seriesCount++)   {     // get the value at the last bar on the chart (if it has been set)     if (Values[seriesCount].IsValidDataPointAt(ChartBars.ToIndex))     {         double plotValue = Values[seriesCount].GetValueAt(ChartBars.ToIndex);           // convert the plot value to the charts "Y" axis point         float chartScaleYValue = chartScale.GetYByValue(plotValue);           // calculate the x and y values for the line to start and end         SharpDX.Vector2 startPoint = new SharpDX.Vector2(startX, chartScaleYValue);         SharpDX.Vector2 endPoint = new SharpDX.Vector2(endX, chartScaleYValue);           // draw a line between the start and end point at each plot using the plots SharpDX Brush color and style         RenderTarget.DrawLine(startPoint, endPoint, Plots[seriesCount].BrushDX,           Plots[seriesCount].Width, Plots[seriesCount].StrokeStyle);           // use the chart control text form to draw plot values along the line         SharpDX.DirectWrite.TextFormat textFormat = chartControl.Properties.LabelFont.ToDirectWriteTextFormat();           // calculate the which will be rendered at each plot using it the plot name and its price         string textToRender = Plots[seriesCount].Name + ": " + plotValue;           // calculate the layout of the text to be drawn         SharpDX.DirectWrite.TextLayout textLayout = new SharpDX.DirectWrite.TextLayout(Core.Globals.DirectWriteFactory,           textToRender, textFormat, 200, textFormat.FontSize);           // draw a line at each plot using the plots SharpDX Brush color at the calculated start point         RenderTarget.DrawTextLayout(startPoint, textLayout, Plots[seriesCount].BrushDX);           // dipose of the unmanaged resources used         textLayout.Dispose();         textFormat.Dispose();     }   } }   protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "OnRender Example";     IsOverlay = true;           AddPlot(Brushes.DarkKhaki, "Open");     AddPlot(Brushes.SeaGreen, "High");     AddPlot(Brushes.Crimson, "Low");     AddPlot(Brushes.DodgerBlue, "Close");   } }   protected override void OnBarUpdate() {   Values[0][0] = Open[0];   Values[1][0] = High[0];   Values[2][0] = Low[0];   Values[3][0] = Close[0]; } | |
| **Navigation:**  [Operations](https://ninjatrader.com/support/helpGuides/nt8/operations.htm) > [Order Entry](https://ninjatrader.com/support/helpGuides/nt8/order_entry.htm) > [Chart Trader](https://ninjatrader.com/support/helpGuides/nt8/chart_trader.htm) >  **Order & Position Display** | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/chart_trader.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/chart_trader.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/collapsed_view.htm) |

**Chart Trader** allows for the placement of orders, and the management of orders and positions, directly from a chart. Orders and positions within **Chart Trader** are displayed in a visual manner, allowing you to quickly compare them with current market movements while modifying orders in real-time. **Chart Trader** contains two primary components: the **Chart Trader** panel, which is used to place, modify, or cancel/close orders and positions, and the chart panel, on which **Chart Trader** draws visual representations of resting orders and open positions.

tog_minus        [Understanding order display](javascript:HMToggle('toggle','UnderstandingOrderDisplay','UnderstandingOrderDisplay_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Order Display**  A resting order is displayed on the chart as a color-coded line connecting an order price label in the right margin of the chart with a second label displaying the order quantity and type.     |  |  | | --- | --- | | Limit Order | Default color is cyan with the text "LMT" | | Stop-Market Order | Default color is pink with text "STP" | | Stop-Limit Order | Default color is violet with text "SLM" | | Market Order | Market orders are not displayed (see **Control Center** [Orders Tab](https://ninjatrader.com/support/helpGuides/nt8/orders_tab.htm) for more information) | | Market if Touched (MIT) Order | Default color is spring green with the text "MIT" | | Stop Loss Order | Default color is red | | Profit Target Order | Default color is lime |     Chart trader color properties can be set via the **Chart Trader** [properties window](https://ninjatrader.com/support/helpGuides/nt8/properties3.htm).    The image below displays how orders are visualized in a NinjaTrader chart with **Chart Trader** enabled.    ChartTrader1    1. Buy stop-limit order for 1 contract at a price of X  2. Buy stop-market order for 1 contract at a price of X  3. Buy limit order for 1 contract at a price of X     |  | | --- | | **Note**: Orders will only display for the selected Account and Instrument. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order__position_display.htm#UnderstandingOrderDisplay)

tog_minus        [Understanding Position Display](javascript:HMToggle('toggle','UnderstandingPositionDisplay','UnderstandingPositionDisplay_ICON'))

|  |  |
| --- | --- |
| **Position Display**  An open position is displayed slightly differently. A position is displayed on the chart as a colored line connecting an entry price label in the right margin of the chart with a second label displaying the position size and current unrealized profit or loss. The text displaying the profit and loss is color coded, with green representing profit and red representing loss. The quantity displayed in the left-hand label is color coded, as well, with green representing a long position and red representing a short position.    **Note**: The display of unrealized PnL in **Chart Trader** can be switched between points, currency, pips, ticks, and percent by either left mouse clicking in the PnL field in the **Chart Trader** panel, or via the **Chart Trader** [Properties](https://ninjatrader.com/support/helpGuides/nt8/properties3.htm) window.    The image below displays the chart with an active position managed by an [Advanced Trade Management](https://ninjatrader.com/support/helpGuides/nt8/advanced_trade_management_atm.htm) strategy.    ChartTrader2    1. Two profit target orders  2. Position size and PnL flag for 2 contracts long  3. Two stop loss orders\*  4. Average entry price  5. PnL in **Chart Trader** panel    \* The stop loss line and flag represents two orders, as indicated by the letter "s" next to the qty number "2."     |  | | --- | | **Note**: Positions will only display for the selected Account and Instrument. | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order__position_display.htm#UnderstandingPositionDisplay)

tog_minus        [Working with multiple instruments](javascript:HMToggle('toggle','WorkingWithMultipleInstruments','WorkingWithMultipleInstruments_ICON'))

|  |
| --- |
| **Multi-Series Charts**  NinjaTrader charts include the ability to plot multiple instruments within a single chart window, and each individual instrument on a chart can be selected and worked with separately using **Chart Trader**. For more information on how to manage instruments on a chart, see the [Working with Multiple Data Series](https://ninjatrader.com/support/helpGuides/nt8/working_with_multiple_data_series.htm) page.    ChartTrader16    In the image above, we have applied a EURUSD instrument, an FDAX ##-## instrument, and a BIDU instrument in three separate panels of the same chart.    **Selecting Data Series**  With more than one instrument applied to a chart, you can change the instrument upon which **Chart Trader** will act via the Instrument dropdown menu. This menu will list all of the **Data Series** currently applied to your chart. When an instrument is selected, only orders and positions for that particular instrument will be displayed in the chart panel, and any quick buttons used or order parameters set in the **Chart Trader** panel will apply to that instrument.    ChartTrader17    1. With the EURUSD selected in the Instruments dropdown menu, we can only that instrument's orders and positions.    ChartTrader18    2. With BIDU selected, we see a different set of orders and positions. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?order__position_display.htm#WorkingWithMultipleInstruments)

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/support/helpGuides/nt8/strategy.htm) > [Order Methods](https://ninjatrader.com/support/helpGuides/nt8/order_methods.htm) >  **Managed Approach** | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/order_methods.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/order_methods.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/advanced_order_handling.htm) |

The Managed approach in NinjaScript is designed to offer the greatest ease of use for beginner to intermediate programmers. The order methods are wrapped in a convenience layer that allows you to focus on your system's trading rules, leaving the underlying mechanics of order management and the relationships between entry orders, exit orders, and positions to NinjaTrader. This approach is best suited for simple to moderate order complexity, and can be further broken down into a Basic/Common Managed approach and a more [Advanced](https://ninjatrader.com/support/helpGuides/nt8/advanced_order_handling.htm) Managed approach. The following section will discuss the use of the Basic/Common approach.

A few key points to keep in mind:

•Orders are submitted as live and working when a strategy is running in real-time

•Profit target, stop loss and trail stop orders are submitted immediately when an entry order is filled, and are tied together via OCO (One Cancels Other)

•Order changes and cancellations are queued in the event that the order is in a state where it can't be cancelled or modified

•By default, orders submitted via Entry() and Exit() methods automatically cancel at the end of a bar if not re-submitted

•Entry() methods will reverse the position automatically. For example if you are in a 1 contract long position and now call EnterShort() -> you will see 2 executions, one to close the prior long position and the other to get you into the desired 1 contract short position.

\* Via the [SetProfitTarget()](https://ninjatrader.com/support/helpGuides/nt8/setprofittarget.htm), [SetStopLoss()](https://ninjatrader.com/support/helpGuides/nt8/setstoploss.htm), [SetTrailStop()](https://ninjatrader.com/support/helpGuides/nt8/settrailstop.htm) and [SetParabolicStop](https://ninjatrader.com/support/helpGuides/nt8/setparabolicstop.htm) methods

tog_minus        [Order submission for entry and exit methods - basic operation](javascript:HMToggle('toggle','OrderSubmissionForEntryAndExitMethodsBasicOperation','OrderSubmissionForEntryAndExitMethodsBasicOperation_ICON'))

|  |  |  |
| --- | --- | --- |
| Orders are primarily submitted from within the [OnBarUpdate()](https://ninjatrader.com/support/helpGuides/nt8/onbarupdate.htm) method when a specific order method is called. By default, orders are kept alive, provided they are re-submitted on each call of the OnBarUpdate() method. If an order is not re-submitted, it is then canceled. Orders can be modified by re-submitting them with changed parameters (a new limit price, for example).    In the example below, a Buy Limit order is working at the bid price, provided that the Close price of the current bar is greater than the current value of the 20 period Simple Moving Average. If the entry condition is no longer true and the order is still active, it will be immediately canceled.     | ns | | --- | | protected override void OnBarUpdate() {     // Entry condition     if (Close[0] > SMA(20)[0])         EnterLongLimit(GetCurrentBid()); } |     This technique allows you the quickest and easiest order submission method suitable for programmers of all levels. Should you want to submit an order and not have to keep re-submitting it to keep it alive you can use an [advanced approach](https://ninjatrader.com/support/helpGuides/nt8/advanced_order_handling.htm) reserved for experienced programmers, which includes an option to keep orders alive until specifically canceled in code. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?managed_approach.htm#OrderSubmissionForEntryAndExitMethodsBasicOperation)

tog_minus        [Order Entry Methods](javascript:HMToggle('toggle','EntryMethods','EntryMethods_ICON'))

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Order Entry Methods** Order entry methods are used to submit orders of different types. Methods exist to submit Market, Market-if-Touched, Limit, Stop Market, and Stop Limit orders. See the order-entry method pages listed in the help guide table of contents under this page for more information on a specific method.    **Signal Names on Entry Methods**  You can optionally tag an entry order with a signal name. Signal names are used to identify executions resulting from the order on a chart and in performance reports. Market positions created from a tagged entry method are marked with the signal name which serves two purposes:    •Used to tie an exit method to a specific position  •Used to identify unique entries in a strategy    Below is an example of placing an Market entry order and an associated Limit exit order, tied together by the signal name of the entry order.     | ns | | --- | | protected override void OnBarUpdate() {   if (CurrentBar < 1) return;     if (Close[0] > Close[1])   {       // Place a Market order to enter long       EnterLong("longEntry");         // Manually place a Profit Target 10 ticks above the current price, tied to the entry order's SignalName       ExitLongLimit(Close[0] + (10 \* TickSize), "longEntry");   } } |       **Defining how Entry Methods are Processed in a Strategy**  You can limit how many entry methods are processed by determining the maximum number of entries in a single direction across all entry methods, or across unique signal names. The following properties can be set in the Strategies window when adding a strategy to a chart or to the Strategies tab of the Control Center window.    •[EntriesPerDirection](https://ninjatrader.com/support/helpGuides/nt8/entriesperdirection.htm) property - Sets the maximum number of entries in a single direction  •[EntryHandling](https://ninjatrader.com/support/helpGuides/nt8/entryhandling.htm) property - Determines if EntriesPerDirection applies across all entries or for entries with specified signal names    The example code below illustrates how the above properties control the processing of entry methods. The code contains two entry conditions and two EnterLong methods, each tagged with unique signal names.     | ns | | --- | | protected override void OnStateChange() {   if (State == State.SetDefaults)   {       EntriesPerDirection = 1;       EntryHandling = EntryHandling.AllEntries;   } }   protected override void OnBarUpdate() {   // Entry condition 1   if (CrossAbove(SMA(10), SMA(20), 1))       EnterLong("Condition 1 Entry");     // Entry condition 2   if (CrossAbove(RSI(14, 3), 30, 1))       EnterLong("Condition 2 Entry"); } |       **Entry Methods on Multi-Instrument Strategies** When running strategies that submit orders to multiple instruments, entry methods will submit orders to the instrument referenced by the [BarsInProgress](https://ninjatrader.com/support/helpGuides/nt8/barsinprogress.htm). The following example assumes that the strategy is running on a 1 minute E-Mini S&P 500 chart. It adds an NQ data series, then enters a position on both instruments.     | ns | | --- | | protected override void OnStateChange() {     AddDataSeries("NQ 09-14", BarsPeriodType.Minute, 1); }   protected override void OnBarUpdate() {     if (BarsInProgress == 0)         EnterLong("ES Trade");     else if (BarsInProgress == 1)         EnterLong("NQ Trade"); } |     More information on using BarsInProgress to filter instruments can be found in the [Advanced Order Handling](https://ninjatrader.com/support/helpGuides/nt8/advanced_order_handling.htm) page. |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?managed_approach.htm#EntryMethods)

tog_minus        [Quantity Type and TIF](javascript:HMToggle('toggle','QuantityTypeAndTif','QuantityTypeAndTif_ICON'))

|  |
| --- |
| You can set the entry order quantity and order type directly in code via the following properties:  •QuantityType - Sets the order quantity is taken from the entry method quantity property or the default strategy quantity size  •[TimeInForce](https://ninjatrader.com/support/helpGuides/nt8/timeinforce.htm) propery - Sets the time in force of the order |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?managed_approach.htm#QuantityTypeAndTif)

tog_minus        [How to close a position](javascript:HMToggle('toggle','HowToCloseAPosition','HowToCloseAPosition_ICON'))

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Closing a Position using a Stop Loss, Trailing Stop and/or Profit Target**  You can predefine a stop loss, trailing stop and/or profit target in a strategy by calling the [SetStopLoss()](https://ninjatrader.com/support/helpGuides/nt8/setstoploss.htm), [SetTrailStop()](https://ninjatrader.com/support/helpGuides/nt8/settrailstop.htm), [SetParabolicStop()](https://ninjatrader.com/support/helpGuides/nt8/setparabolicstop.htm) or [SetProfitTarget()](https://ninjatrader.com/support/helpGuides/nt8/setprofittarget.htm) methods from inside the [OnStateChange()](https://ninjatrader.com/support/helpGuides/nt8/onstatechange.htm) event handler. When these methods are called, they submit live working orders in real-time as executions are reported as a result of calling an entry method. These orders are also tied via OCO (One Cancels Other).    Stop losses and profit target can be generated for each fill or each position. This is determined by the "Stop & target submission" property which is set in the Strategies window. Possible values are listed below:    **ByStrategyPosition** - When this is selected, only one stop loss, trail stop and/or profit target order is submitted. As entry executions come in, the order size is amended. The downside of this approach is that if you receive partial fills, the orders are re-inserted into the exchange order queue. The upside is that if you broker charges you commission per order (not per quantity), you will not incur additional commission expenses.    **PerEntryExecution** - When this is selected, a stop loss, trail stop and/or profit target order is submitted for each partial fill received. The downside is that if your broker charges commission per order, you can incur very expensive commission costs if you receive partial fills. The upside is that orders are submitted as soon as possible, giving you the advantage of getting into the order queue immediately.      **Closing a Position using an Exit Method**  Exit methods submit orders to close out a position in whole or in part. When closing a position with Exit orders, the order quantity will be reduced as the strategy position reduces - for example, if we use [ExitLongStopMarket()](https://ninjatrader.com/support/helpGuides/nt8/exitlongstopmarket.htm) and [ExitLongStopLimit()](https://ninjatrader.com/support/helpGuides/nt8/exitlongstoplimit.htm) to protect a position and one of those orders gets filled, the other order associated with exiting that position will reduce their quantity.    As with entry methods, more information about specific exit methods can be found in this Help Guide's table of contents, beneath this page.      **Closing a Partial Position using an Exit Method**  You can close out a partial position by specifying the exit quantity. The following example first enters long for three contracts. Then, each subsequent bar update submits a market order to exit one contract until the position is completely closed. "ExitLong(1)" will be ignored if a long market position does not exist.     | ns | | --- | | protected override void OnBarUpdate() {     if (CrossAbove(SMA(10), SMA(20), 1))         EnterLong(3);       ExitLong(1); } |       **FromEntrySignal -- Using Signal Names in Exit Methods**  Identifying entries with a signal name allows you to place multiple unique entries within a single strategy and call exit methods with specified signal names, so that only a position created with the specified signal name is closed. In the example below, there are two entry conditions which create positions, and two exit conditions specifying which position to close based on the signal name.     | ns | | --- | | protected override void OnBarUpdate() {     // Entry condition 1     if (CrossAbove(SMA(10), SMA(20), 1))         EnterLong("Condition 1 Entry");       // Entry condition 2     if (CrossAbove(RSI(14, 3), 30, 1))         EnterLong("Condition 2 Entry");       // Closes the position created by entry condition 1     if (CrossBelow(SMA(10), SMA(20), 1))         ExitLong("Condition 1 Entry");       // Closes the position created by entry condition 2     if (CrossBelow(RSI(14, 3), 70, 1))         ExitLong("Condition 2 Entry"); } |        |  | | --- | | **Tip**:  If you do not specify a "fromEntrySignal" parameter the entire position is exited rendering your strategy flat. |        | ns | | --- | | protected override void OnBarUpdate() {     if (Position.MarketPosition == MarketPosition.Flat)   {     // Entry condition 1     if (CrossAbove(SMA(10), SMA(20), 1))         EnterLong("Condition 1 Entry");   }     if (Position.MarketPosition != MarketPosition.Flat)   {     // Scale in condition 2 for position management     if (CrossAbove(RSI(14, 3), 30, 1))         EnterLong("Condition 2 Entry");       // Exit all positions using an empty string (could also use string.Empty)     if (CrossBelow(SMA(10), SMA(20), 1))         ExitLong("Exit All", "");     } } | |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?managed_approach.htm#HowToCloseAPosition)

tog_minus        [Understanding core order objects](javascript:HMToggle('toggle','Understandingcoreorderobjects','Understandingcoreorderobjects_ICON'))

|  |
| --- |
| When using order methods such as [EnterLong()](https://ninjatrader.com/support/helpGuides/nt8/enterlong.htm), [ExitShortLimit()](https://ninjatrader.com/support/helpGuides/nt8/exitshortlimit.htm), etc, a direct [order object](https://ninjatrader.com/support/helpGuides/nt8/order.htm) is returned for the NinjaTrader Core.  These objects can be used throughout the lifetime of your strategy to provide additional metadata concerning your strategy, as well as apply advanced concepts such as [CancelOrder()](https://ninjatrader.com/support/helpGuides/nt8/managed_cancelorder.htm) and [ChangeOrder()](https://ninjatrader.com/support/helpGuides/nt8/managed_changeorder.htm).  More information about this advanced concept which is discussed under the [Advanced Order Handling](https://ninjatrader.com/support/helpGuides/nt8/advanced_order_handling.htm) section |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?managed_approach.htm#Understandingcoreorderobjects)

tog_minus        [Internal Order Handling Rules that Reduce Unwanted Positions](javascript:HMToggle('toggle','InternalOrderHandlingRulesThatReduceUnwantedPositions','InternalOrderHandlingRulesThatReduceUnwantedPositions_ICON'))

|  |  |  |
| --- | --- | --- |
| To prevent situations in real-time in which you may have multiple orders working to accomplish the same task, there are some "under the hood" rules that a NinjaScript strategy follows when Managed order methods are called. For example, if your strategy had a limit order for 1 contract working as a Profit Target, but then your strategy was also programmed to reverse the position at the price very close to the target limit order, then submitting both orders can be risky, since it could lead to a larger position than the strategy is designed to enter if both orders got filled in quick succession by the exchange.     |  | | --- | | **Note**: These rules do not apply to market orders, such as ExitLong() or ExitShort(). |     For the most part, you do not need to be intimately familiar with these rules as you develop your strategies. It is all taken care of for you internally within a strategy. If a rule is violated, you will be notified through an error log in the Control Center Log tab.     |  | | --- | | **Note**:  To prevent excessive logging which could degrade performance, you will only be notified of the very first order which has violated an order handling rule. Subsequent orders which violate a rule will not be notified through the error log. |     The following rules are true*per unique signal name*:    Methods that generate orders to **enter** a position will be ignored if:  •A position is open and an order submitted by a non market order exit method ([ExitLongLimit()](https://ninjatrader.com/support/helpGuides/nt8/exitlonglimit.htm) for example) is active and the order is used to open a position in the opposite direction  •A position is open and an order submitted by a set method ([SetStopLoss()](https://ninjatrader.com/support/helpGuides/nt8/setstoploss.htm) for example) is active and the order is used to open a position in the opposite direction  •A position is open and two or more Entry methods to reverse the position are entered together. In this case the second Entry order will be ignored.  •The strategy position is flat and an order submitted by an enter method ([EnterLongLimit()](https://ninjatrader.com/support/helpGuides/nt8/enterlonglimit.htm) for example) is active and the order is used to open a position in the opposite direction  •The entry signal name is not unique    Methods that generate orders to **exit** a position will be ignored if:  •A position is open and an order submitted by an enter method ([EnterLongLimit()](https://ninjatrader.com/support/helpGuides/nt8/enterlonglimit.htm) for example) is active and the order is used to open a position in the opposite direction  •A position is open and an order submitted by a set method ([SetStopLoss()](https://ninjatrader.com/support/helpGuides/nt8/setstoploss.htm) for example) is active    Set() methods that generate orders to **exit** a position will be ignored if:  •A position is open and an order submitted by an enter method ([EnterLongLimit()](https://ninjatrader.com/support/helpGuides/nt8/enterlonglimit.htm) for example) is active and the order is used to open a position in the opposite direction  •A position is open and an order submitted by a non market order exit method ([ExitLongLimit()](https://ninjatrader.com/support/helpGuides/nt8/exitlonglimit.htm) for example) is active |

[permalink](https://ninjatrader.com/support/helpGuides/nt8/index.html?managed_approach.htm#InternalOrderHandlingRulesThatReduceUnwantedPositions)

|  |  |
| --- | --- |
| [Advanced Order Handling](https://ninjatrader.com/support/helpGuides/nt8/advanced_order_handling.htm) | Through advanced order handling you can submit, change and cancel orders at your discretion through any event-driven method within a strategy. |
| [CancelOrder()](https://ninjatrader.com/support/helpGuides/nt8/managed_cancelorder.htm) | Cancels a specified order. |
| [ChangeOrder()](https://ninjatrader.com/support/helpGuides/nt8/managed_changeorder.htm) | Amends a specified [Order](https://ninjatrader.com/support/helpGuides/nt8/order.htm). |
| [EnterLong()](https://ninjatrader.com/support/helpGuides/nt8/enterlong.htm) | Generates a buy market order to enter a long position. |
| [EnterLongLimit()](https://ninjatrader.com/support/helpGuides/nt8/enterlonglimit.htm) | Generates a buy limit order to enter a long position. |
| [EnterLongMIT()](https://ninjatrader.com/support/helpGuides/nt8/enterlongmit.htm) | Generates a buy MIT order to enter a long position. |
| [EnterLongStopLimit()](https://ninjatrader.com/support/helpGuides/nt8/enterlongstoplimit.htm) | Generates a buy stop limit order to enter a long position. |
| [EnterLongStopMarket()](https://ninjatrader.com/support/helpGuides/nt8/enterlongstopmarket.htm) | Generates a buy stop market order to enter a long position. |
| [EnterShort()](https://ninjatrader.com/support/helpGuides/nt8/entershort.htm) | Generates a sell short market order to enter a short position. |
| [EnterShortLimit()](https://ninjatrader.com/support/helpGuides/nt8/entershortlimit.htm) | Generates a sell short stop limit order to enter a short position. |
| [EnterShortMIT()](https://ninjatrader.com/support/helpGuides/nt8/entershortmit.htm) | Generates a sell MIT order to enter a short position. |
| [EnterShortStopLimit()](https://ninjatrader.com/support/helpGuides/nt8/entershortstoplimit.htm) | Generates a sell short stop limit order to enter a short position. |
| [EnterShortStopMarket()](https://ninjatrader.com/support/helpGuides/nt8/entershortstopmarket.htm) | Generates a sell short stop order to enter a short position. |
| [ExitLong()](https://ninjatrader.com/support/helpGuides/nt8/exitlong.htm) | Generates a sell market order to exit a long position. |
| [ExitLongLimit()](https://ninjatrader.com/support/helpGuides/nt8/exitlonglimit.htm) | Generates a sell limit order to exit a long position. |
| [ExitLongMIT()](https://ninjatrader.com/support/helpGuides/nt8/exitlongmit.htm) | Generates a sell MIT order to exit a long position. |
| [ExitLongStopLimit()](https://ninjatrader.com/support/helpGuides/nt8/exitlongstoplimit.htm) | Generates a sell stop limit order to exit a long position. |
| [ExitLongStopMarket()](https://ninjatrader.com/support/helpGuides/nt8/exitlongstopmarket.htm) | Generates a sell stop market order to exit a long position. |
| [ExitShort()](https://ninjatrader.com/support/helpGuides/nt8/exitshort.htm) | Generates a buy to cover market order to exit a short position. |
| [ExitShortLimit()](https://ninjatrader.com/support/helpGuides/nt8/exitshortlimit.htm) | Generates a buy to cover limit order to exit a short position. |
| [ExitShortMIT()](https://ninjatrader.com/support/helpGuides/nt8/exitshortmit.htm) | Generates a buy to cover MIT order to exit a short position. |
| [ExitShortStopLimit()](https://ninjatrader.com/support/helpGuides/nt8/exitshortstoplimit.htm) | Generates a buy to cover stop limit order to exit a short position. |
| [ExitShortStopMarket()](https://ninjatrader.com/support/helpGuides/nt8/exitshortstopmarket.htm) | Generates a buy to cover stop market order to exit a short position. |
| [GetRealtimeOrder()](https://ninjatrader.com/support/helpGuides/nt8/getrealtimeorder.htm) | Returns a matching real-time order object based on a specified historical order object reference. |
| [SetParabolicStop()](https://ninjatrader.com/support/helpGuides/nt8/setparabolicstop.htm) | Generates a parabolic type trail stop order with the signal name "Parabolic stop" to exit a position. |
| [SetProfitTarget()](https://ninjatrader.com/support/helpGuides/nt8/setprofittarget.htm) | Generates a profit target order with the signal name "Profit target" to exit a position. |
| [SetStopLoss()](https://ninjatrader.com/support/helpGuides/nt8/setstoploss.htm) | Generates a stop loss order with the signal name "Stop loss" used to exit a position. |
| [SetTrailStop()](https://ninjatrader.com/support/helpGuides/nt8/settrailstop.htm) | Generates a trail stop order with the signal name "Trail stop" to exit a position. |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/support/helpGuides/nt8/common.htm) > [Analytical](https://ninjatrader.com/support/helpGuides/nt8/market_data.htm) >  **Slope()** | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/most_recent_occurence_mro.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/market_data.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/ticksize.htm) |

**Definition**

Returns a measurement of the steepness of a price series (y value) measured by the change over time (x value).  The return value can also be thought of as the ratio between the startBarsAgo and endBarsAgo parameters passed to the method.

The formula which is returned from the parameters passed is:

**(series[endBarsAgo] - series[startBarsAgo]) / (startBarsAgo - endBarsAgo)**

|  |
| --- |
| **Note**:  The return value should **NOT** be confused with the angle (or radians) of a line that displays on the chart. |

**Method Return Value**

This method returns a double value indicating the slope of a line;  A value of 0 returns if the either the startBars or endBars parameters are less than 0 or both parameters are of equal value.

**Syntax**  
Slope(ISeries<double> series, int startBarsAgo, int endBarsAgo)

|  |
| --- |
| **Warning**:  The "startBarsAgo" parameter **MUST** be greater than the "endBarsAgo" parameter |

**Parameters**

|  |  |
| --- | --- |
| series | Any Series<double> type object such as an indicator, Close, High, Low, etc... |
| startBarsAgo | The starting point of a series to be evaluated |
| endBarsAgo | The ending point of a series to be evaluated |

|  |
| --- |
| **Tip**: Thinking in degrees, for example a 1 to -1 return range would translate to 45 to -45. To convert you could look into working with this formula - Math.Atan(Slope) \* 180 / Math.PI |

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Prints the slope of the 20 period simple moving average of the last 10 bars   Print(Slope(SMA(20), 10, 0));   } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/support/helpGuides/nt8/strategy.htm) >  **AddChartIndicator()** | | [Previous page](https://ninjatrader.com/support/helpGuides/nt8/strategy_account.htm) [Return to chapter overview](https://ninjatrader.com/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/support/helpGuides/nt8/addperformancemetric.htm) |

**Definition**

Adds an indicator to the strategy only for the purpose of displaying it on a chart.

|  |
| --- |
| **Notes**:  •Only the Plot properties of an indicator added by AddChartIndicator() will be accessible in the Indicators dialogue on charts. Other properties must be set in code.  •To add Bars objects to your strategy for calculation purposes see the [AddDataSeries()](https://ninjatrader.com/support/helpGuides/nt8/adddataseries.htm) method.  •An indicator being added via AddChartIndicator() cannot use any additional data series hosted by the calling strategy, but can only use the strategy's primary data series. If you wish to use a different data series for the indicator's input, you can add the series in the indicator itself and explicitly reference it in the indicator code (please make sure though the hosting strategy has the same [AddDataSeries()](https://ninjatrader.com/support/helpGuides/nt8/adddataseries.htm) call included as well)  o If a secondary or null Bars series is specified by the calling strategy (not the indicator itself), the strategy's primary series will be substituted instead.  •Dynamically using [DrawOnPricePanel](https://ninjatrader.com/support/helpGuides/nt8/drawonpricepanel.htm) in an indicator outside of State.SetDefaults may show issues when working with that indicator through a hosting strategy via [AddChartIndicator()](https://ninjatrader.com/support/helpGuides/nt8/addchartindicator.htm). |

**Method Return Value**

This method does not return a value.

**Syntax**  
AddChartIndicator(IndicatorBase indicator)

|  |
| --- |
| **Warning**:  This method should **ONLY** be called from the [OnStateChange()](https://ninjatrader.com/support/helpGuides/nt8/onstatechange.htm) method during **State.DataLoaded** |

**Parameters**

|  |  |
| --- | --- |
| indicator | An indicator object |

**Examples**

| ns |
| --- |
| protected override void OnStateChange() {     if (State == State.DataLoaded)     {         // Charts a 20 period simple moving average to the chart         AddChartIndicator(SMA(20));     } } |

|  |
| --- |
| **Tip**:  If you are adding an indicator which is dependent on the correct [State](https://ninjatrader.com/support/helpGuides/nt8/state.htm) of the indicator, you will need to ensure that you are also calling the indicator from the strategy in [OnBarUpdate()](https://ninjatrader.com/support/helpGuides/nt8/onbarupdate.htm), otherwise your indicator will only process in **State.RealTime** for performance optimizations. |

| ns |
| --- |
| protected override void OnStateChange() {   if (State == State.DataLoaded)   {     // Charts a 20 period simple moving average to the chart     AddChartIndicator(SMA(20));   } }   protected override void OnBarUpdate() {     // call SMA() historically to ensure the indicator processes its historical states as well   double sma = SMA(20)[0]; } |