|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Educational Resources](https://ninjatrader.com/es/support/helpGuides/nt8/educational_resources.htm) > [AddOn Development Overview](https://ninjatrader.com/es/support/helpGuides/nt8/addon_development_overview.htm) >  **Other Uses for an AddOn** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/creating_your_own_addon_window.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/addon_development_overview.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/c_method_functions_reference.htm) |

**Modifying Existing NinjaTrader Windows**

To modify an existing type of NinjaTrader window (for example, to add a button to all charts), you will first need to obtain a reference to each individual window of that type that is open. This can be done by overriding the OnWindowCreated() method, then declaring an object of the Type of the window you are looking for, and finally assigning the object a reference to the Window passed into the method:

| ns |
| --- |
| *// OnWindowCreated() will be called any time a new NTWindow is created. It will be called in the thread of that window* **protected** **override** **void** OnWindowCreated(Window window) {   *// Declare a Chart object and instantiate it to the Window passed into the method*   Gui.Chart.Chart myChart = window **as** Gui.Chart.Chart;       *// Use this check to return if the calling Window is not of the Type you are looking for*   **if** (myChart == **null**)       **return**; } |

If you are unsure of the Type name for a particular type of window, you can open an instance of that window then run the code below, which will print the Type to the Output Window:

| ns |
| --- |
| **protected** **override** **void** OnWindowCreated(Window window) {   *// Print the Type of any open windows, for future reference*   Print(window.ToString()); } |

Once you've obtained a reference to a window, you can then directly manipulate the WPF grids, controls, and other elements to customize its user interface or functionality. For example, if your goal was to add a new button to Chart Trader on all charts, you could use your reference to Chart objects to first locate their attached Chart Trader instances, then place a custom-defined button directly into the WPF grid used to lay out buttons in Chart Trader. Since this code would run within OnWindowCreated(), it would be applied to every Chart Trader instance that is open. You would not be changing the format used to create Chart Traders in the first place, but would rather be detecting every open instance and adding the buttons into them. This is an important distinction to make, because this approach requires that you also remove the elements you've added when each window is destroyed.

| ns |
| --- |
| *// Declare a Chart, ChartTrader, and UI elements to add to Chart Trader* Gui.Chart.Chart myChart; Gui.Chart.ChartTrader chartTrader; Button sampleButton; Grid myGrid; Grid mainGrid;   **protected** **override** **void** OnWindowCreated(Window window) {   *// Instantiate myChart by assigning a reference to the calling Window*   myChart = window **as** Gui.Chart.Chart;     **if** (myChart == **null**)   {       return;   }     *//find chart trader from myChart's Chart Control by its Automation ID: "ChartWindowChartTrader"*   chartTrader = Window.GetWindow(myChart.ActiveChartControl.Parent).FindFirst("ChartWindowChartTraderControl") **as** Gui.Chart.ChartTrader;     **if** (chartTrader == **null**)   {       **return**;   }     *// Instantiate sampleButton*   sampleButton = **new** Button   {       Content = "Sample Button",       Style = System.Windows.Application.Current.TryFindResource("Button") **as** Style   };     *// Attach a custom event handler to the .Click event*   sampleButton.Click += SampleButton\_Click;     *// Set a custom AutomationId for the button, so that it can be referenced elsewhere the same way we found Chart Trader*   System.Windows.Automation.AutomationProperties.SetAutomationId(sampleButton, "SampleButton");     *//this is the main chart trader grid where the default buttons and controls reside*   mainGrid = chartTrader.FindName("grdMain") **as** Grid;     *// Return if Chart Trader is null*   **if** (mainGrid == **null**)   {       **return**;   }     *// by default, there will be 7 rows in Chart Trader, we need to add a new row for the new button*   **if** (mainGrid.RowDefinitions.Count <= 7)       mainGrid.RowDefinitions.Add(**new** RowDefinition());     *//define a new grid, and add our button to that grid*   myGrid = **new** Grid();   myGrid.Children.Add(sampleButton);     *//set my grid to the new row*   Grid.SetRow(myGrid, 8);     *//finally, add our grid to the main grid*   mainGrid.Children.Add(myGrid); }   **private** **void** SampleButton\_Click(**object** sender, RoutedEventArgs e) {   Print("Sample Button Clicked"); } |

Since we are dynamically adding elements to open windows, it is important to clean up any unused resources and detach any event handlers when the affected windows are destroyed. You can use the same approach as shown above to obtain a reference to each affected window within the OnWindowDestroyed() method:

| ns |
| --- |
| **protected** **override** **void** OnWindowDestroyed(Window window) {   *// Return if there is no button, or if the destroyed window is not a chart*   **if**(sampleButton == **null** || !(window **is** Gui.Chart.Chart))   {       **return**;   }     *// Detach the event handler from the .Click event, remove the grid, and nullify the button*   sampleButton.Click -= SampleButton\_Click;   mainGrid.Children.Remove(myGrid);   sampleButton = **null**; } |

Below is another example of adding elements into chart windows. In this example, we add a new panel to the top of all chart windows, then take all existing chart content and move it into a row beneath the panel we've just added:

| ns |
| --- |
| **protected** **override** **void** OnWindowCreated(Window window) {   *// Obtain a reference to any chart that triggered OnWindowCreated*   Chart Window = window **as** Chart;       *// Instantiate a grid to hold a reference to the content of the chart window*   Grid mainWindowGrid = Window.Content **as** Grid;       *// Add existing row definition for existing row if it is not present*   **if** (mainWindowGrid.RowDefinitions.Count == 0)   {       mainWindowGrid.RowDefinitions.Add(**new** RowDefinition());   }                   *// Instantiate a RowDefinition and set its height*   RowDefinition row = **new** RowDefinition();   row.Height = **new** GridLength(PanelLength);       *// Insert the new row into the chart's main window grid*   mainWindowGrid.RowDefinitions.Insert(0, row);       *//Move Existing Elements down one row, since our new content will take the top row*   **foreach** (UIElement element **in** mainWindowGrid.Children)   {       element.SetValue(Grid.RowProperty, (**int**)element.GetValue(Grid.RowProperty) + 1);   }       *//Create the Top Panel grid and add it to our newly defined row*   Grid Panel = **new** Grid();   Panel.SetValue(Grid.RowProperty, 0);   mainWindowGrid.Children.Add(Panel);     *//Create a sample text block and add it to the Top/Bottom Panel Grid.*   TextBlock TextBlock = **new** TextBlock();   TextBlock.Text = PanelDirection.ToString() + " Panel (" + PanelLocation.ToString() + ") Sample Text Block";   TextBlock.Foreground = Brushes.Red;   TextBlock.SetValue(Grid.RowProperty, 0);   Panel.Children.Add(TextBlock); } |

**Accessing Account Data**

From time to time, you may need to access certain global data, such as account values, order states, position info, etc. In these cases, you can subscribe to an appropriate event using a custom event handler method. Below is a list of a few such events which can be captured:

|  |  |
| --- | --- |
| <Account>.[AccountItemUpdate](https://ninjatrader.com/es/support/helpGuides/nt8/accountitemupdate.htm) | Triggers on account item updates |
| <Account>.[ExecutionUpdate](https://ninjatrader.com/es/support/helpGuides/nt8/executionupdate.htm) | Triggers on any execution |
| <Account>.[OrderUpdate](https://ninjatrader.com/es/support/helpGuides/nt8/orderupdate.htm) | Triggers on any order state changes |
| <Account>.[PositionUpdate](https://ninjatrader.com/es/support/helpGuides/nt8/positionupdate.htm) | Triggers on any position updates |

| ns |
| --- |
| *// Custom Subscribe() method to refresh subscriptions* **private** **void** Subscribe() {   **if** (myAccount != **null**)   {       *// Unsubscribe to any prior account subscriptions*       myAccount.AccountItemUpdate -= OnAccountItemUpdate;       myAccount.ExecutionUpdate -= OnExecutionUpdate;       myAccount.OrderUpdate -= OnOrderUpdate;       myAccount.PositionUpdate -= OnPositionUpdate;         *// Subscribe to new account subscriptions*       myAccount.AccountItemUpdate   += OnAccountItemUpdate;       myAccount.ExecutionUpdate     += OnExecutionUpdate;       myAccount.OrderUpdate         += OnOrderUpdate;       myAccount.PositionUpdate     += OnPositionUpdate;   } }    **private** **void** OnAccountItemUpdate(**object** sender, AccountItemEventArgs e) {   *// Handle account item updates* }   **private** **void** OnExecutionUpdate(**object** sender, AccountItemEventArgs e) {   *// Handle execution updates* }   **private** **void** OnOrderUpdate(**object** sender, AccountItemEventArgs e) {   *// Handle order updates* }   **private** **void** OnPositionUpdate(**object** sender, AccountItemEventArgs e) {   *// Handle position updates* } |

**Accessing Market Data**

Market data can be accessed via a BarsRequest object, which can provide real-time or snapshot data for use by your classes. A BarsRequest object can be loaded with a series of bar data without the need to actually draw bars on a chart. The BarsRequest object can then be accessed via the BarsUpdateEventArgs object passed into your event handler via the BarsRequest's Update method. The process for using a BarsRequest is as follows:

1.Instantiate an Instrument object

2.Instantiate and parameterize a BarsRequest object

3.Hook the BarsRequest's Update event to a custom event handler

4.Call the BarsRequest's Request() method

5.Access bars data directly from the BarsRequest object within your event handler method

| ns | |
| --- | --- |
| *// Custom method to perform a BarsRequest* **private** NinjaTrader.Data.BarsRequest DoBarsRequest(Instrument instrument, **int** lookBackPeriod) {   *// Declare a BarsRequest object*   NinjaTrader.Data.BarsRequest barsRequest;       *// Request x number of days back of data.*   barsRequest = **new** NinjaTrader.Data.BarsRequest(instrument, DateTime.Now.AddDays(-lookBackPeriod), DateTime.Now);     *// If you wish to request x number of bars back instead you can use this signature:*   *// barsRequest = new NinjaTrader.Data.BarsRequest(instrument, lookBackPeriod);*         *// Parameterize the request*   barsRequest.BarsPeriod = **new** NinjaTrader.Data.BarsPeriod { BarsPeriodType = BarsPeriodType.Minute, Value = 60 };   barsRequest.TradingHours     = NinjaTrader.Data.TradingHours.Get("Default 24 x 7");       *// Additional parameters which could be set*   *// barsRequest.IsDividendAdjusted      = true;*   *// barsRequest.IsResetOnNewTradingDay   = false;*   *// barsRequest.IsSplitAdjusted         = true;*   *// barsRequest.LookupPolicy            = LookupPolicies.Provider;*   *// barsRequest.MergePolicy            = MergePolicy.DoNotMerge;*    *// Attach event handler for real-time events if you want to process real-time data*   barsRequest.Update     += MyOnBarUpdate;       *// Call the Request method on the BarsRequest object to request the bars*   barsRequest.Request(**new** Action<NinjaTrader.Data.BarsRequest, ErrorCode, **string**>((bars, errorCode, errorMessage) =>   {       Dispatcher.InvokeAsync(**new** Action(() =>       {           **if** (errorCode != ErrorCode.NoError)           {               *// Handle any errors in requesting bars here*               outputBox.Text = **string**.Format("Error on requesting bars: {0}, {1}", errorCode, errorMessage);               **return**;           }       }));   }));       *// Return the Bars Request to any callers of this method*   **return** barsRequest; }   // BarsUpdateEventArgs is provided by the BarsRequest's Update event private void MyOnBarUpdate(object sender, NinjaTrader.Data.BarsUpdateEventArgs e) {   /\* Dispatcher.InvokeAsync() is needed for multi-threading considerations. When processing events outside of the UI thread, and we want to    influence the UI .InvokeAsync() allows us to do so. It can also help prevent the UI thread from locking up on long operations. \*/   Dispatcher.InvokeAsync(() =>   {       /\* Depending on the BarsPeriod type of your barsRequest you can have situations where more than one bar is updated by a single tick        Be sure to process the full range of updated bars to ensure you did not miss a bar. \*/         // Process updated bars on each tick       for (int i = e.MinIndex; i <= e.MaxIndex; i++)       {           // Processing every single tick           outputBox.Text = string.Format("REALTIME BARS{0}Time: {1}{0}Open: {2}{0}High: {3}{0}Low: {4}{0}Close: {5}",               Environment.NewLine,               e.BarsSeries.GetTime(i),               e.BarsSeries.GetOpen(i),               e.BarsSeries.GetHigh(i),               e.BarsSeries.GetLow(i),               e.BarsSeries.GetClose(i));       }       );   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) >  **Code Breaking Changes** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript_best_practices.htm) |

The following document is intended as a high level overview of the NinjaScript changes you can expect between NinjaTrader 7 and NinjaTrader 8.  For specific information on a particular method or property, you can refer to the dynamically formatted **Code Breaking table** at the bottom of this page.  We recommend using the **Filter** and **Sorting** features built into the table, as well checking the **Summary** column and expanding the **Details** section of each entry for general information.  Referring to the conveniently linked NinjaTrader 8 and NinjaTrader 7 documentation will provide specific information on syntax, usage, and examples of any new implementation or element names.

|  |
| --- |
| **Note**:  Information on this page focuses on **supported** **(documented)** NinjaTrader methods and properties shared between versions.  NinjaTrader 8 has seen a significant increase in supported NinjaTrader code, however if you were using previously **undocumented** NinjaTrader 7 methods or properties, they will **NOT** be covered in this topic.  You may be able to find more information on previously **undocumented** methods and properties in the NinjaTrader 8 Help Guide, or our support staff will also be happy to personally point you in the right direction. |

|  |
| --- |
| **Critical**:   If your product uses **unsupported (undocumented)** elements we strongly urge you to put your scripts through thorough testing to ensure they still behave as expected.  There is **NO** guarantee that previously **undocumented** method or property behavior has not changed in the new version of NinjaTrader 8. |

For questions or comments, please contact us at platformsupport@ninjatrader.com

tog_minus        [Implementation Changes Overview](javascript:HMToggle('toggle','ImplementationChangesOverview','ImplementationChangesOverview_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Initialize(), OnStartUp(), OnTermination()**  NinjaTrader 8 has simplified the methods used to set or release various resources during the lifetime of a NinjaTrader object to a single [**OnStateChange()**](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method. This single method is guaranteed to be called for every change in **State** of the object.  It is from this method you can monitor the progression of the object throughout its lifetime in order to setup various resources, set properties, or take action the moment **State** has changed.  This method also exposes a [**State**](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm) variable which can be used in various other methods, such as**OnBarUpdate(),** in order to tell your indicator or strategy to process data depending on the actual **State** of the object.    For example, pushing settings to the UI, or setting initial values for public properties can now be done use **OnStateChange()** when the state has reached**State.SetDefaults**:     | ns | | --- | | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     // set the default properties     Name = "My Indicator";     Fast = 10;     Slow = 25;     IsOverlay = true;     IsAutoScale = true;   } } |       If you have custom resources that need to be setup before the NinjaTrader object is active and processing data, instead of using the**Initialize()** method, you can now set this up once the **OnStateChange()** method has reached **State.Configure** state:     | ns | | --- | | protected override void OnStateChange() {   if (State == State.Configure)   {     // Add a 5 minute Bars object to the strategy     AddDataSeries(Data.BarsPeriodType.Minute, 5);     // setup a custom data series     spread = new Series<double>(this);     // setup a 20-period EMA indicator     ema = EMA(20);     // add indicator to strategy for visual purposes     AddChartIndicator(ema);     } } |       NinjaTrader 7 had no concept to detect when your NinjaTrader object was transitioning from processing Historical data to processing Real-time data.  Now with NinjaTrader 8, the **OnStateChange()** method provides a **State.Transition** state which will notify you when this change is about to occur.  If your NinjaTrader 7 indicators or strategies were using custom methods to try to detect this transition, your custom methods may be refactored under this new state:     | ns | | --- | | protected override void OnStateChange() {   if (State == State.Transition)   {     Print("We're going to real-time data...");     // setup your real-time data resources here   } } |       When your NinjaTrader object is shutting down, and you need clean up any custom device resources, instead of using **OnTermination()**, you should now clean up these resources once the **OnStateChange()** method has reached the **State.Terminated** state:     | ns | | --- | | protected override void OnStateChange() {   if (State == State.Terminated)   {     // release any device resources     if(myTimer != null)         myTimer = null;   } } |     NinjaTrader previously used a **Historical** bool property to notify when an indicator or strategy bar was being processed historically or real-time.  The NinjaTrader 8 **OnStateChange()** approach has now introduced a class level variable **State** where you can check for **State.Historical** or**State.Realtime**in any of the other event methods which will allow you to take action depending on the desired state:     | ns | | --- | | protected override void OnBarUpdate() {   // only process on real-time data   if (State == State.Historical)     return;     else if (State >= State.Realtime)       // rest of logic } |     **Strategies, Orders, and Accounts**  Low level access has been provided to allow more flexibility with the information pertaining to trade data.    •IOrders, IExecution, and IPosition interfaces have all been replaced directly with the corresponding object  •The signatures of the related NinjaScript events have changed to match the NinjaTrader internal Update events  •Methods now return and update with the object instance generated, instead of the previously used interface     |  | | --- | | **Tip**:  Since NinjaTrader 8 now exposes the direct **Order** object, rather than an **IOrder** interface, it is possible to receive **null object reference errors** if you attempt to access an order object before the entry or exit order method has returned.  To prevent these situations, it is recommended to assign your strategies **Order** variables in the **OnOrderUpdate()** method and match them by their **signal name** (order.Name).  Please see the example beginning on line #22 below for demonstration of assigning order objects to private variables. |      | ns | | --- | | Order myOrder = null;   protected override void OnBarUpdate() {           if (Position.MarketPosition == MarketPosition.Flat && myOrder == null)     EnterLongLimit(Low[0], "Entry");     if (myOrder != null)   {     Print(myOrder.OrderState);           if (myOrder.OrderState == OrderState.Cancelled || myOrder.OrderState == OrderState.Filled)         myOrder = null;               } }         protected override void OnOrderUpdate(Cbi.Order order, double limitPrice, double stopPrice,   int quantity, int filled, double averageFillPrice,   Cbi.OrderState orderState, DateTime time, Cbi.ErrorCode error, string comment) {         // compare the order object created via EnterLongLimit by the signal name   if (myOrder == null && order.Name == "Entry")   {     // assign myOrder to matching order update     myOrder = order;           } } |     **Data Series**  Previously there had been type specific Data Series implementations (e.g., IntSeries, TimeSeries, BoolSeries, etc).  Now there just is a template [Series<T>](https://ninjatrader.com/es/support/helpGuides/nt8/seriest.htm) class which could be used generically and even allows support for additional types:     | ns | | --- | | Series<double> mySeries = new Series<double>(this); Series<DateTime> myTimeSeries = new Series<DateTime>(this); |     The **DataSeries.Set()** method used to assign Data Series or Plot values has been removed and values can now be stored using a single assignment operator:     | ns | | --- | | protected override void OnBarUpdate() {   // set public plotting data series close value of current bar   MyPlot[0] = Close[0];   // set custom Series<DateTime> to time value of current bar   myTimeSeries[0] = Time[0];         } |     **Drawing**  The DrawObjects used in NinjaTrader have received a number of changes:    •All DrawObjects have been moved to a separate **NinjaScript.DrawingTools** namespace and are properly known as **DrawingTools**  •Drawing Methods called from indicators or strategies have been moved to a new static partial **Draw** class  •Drawing Methods have all received a signature change which requires you specify the owner (object) which drew the **DrawingTool** object  •Drawing Methods no longer returns an interface but rather an instance of the **DrawingTool** object itself  •Drawing Methods now use the [System.Windows.Media.Brushes](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx) class instead of the [System.Drawing.Color](https://msdn.microsoft.com/en-us/library/system.drawing.color(v=vs.110).aspx) structure     |  | | --- | | **Tip**:  DrawingTools are now completely unprotected and you can review their source code from the DrawingTools folder of the NinjaScript Editor's explorer menu |      | ns | | --- | | // example syntax Draw.Line(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush)   // example usage Draw.Line(this, "tag1", true, 10, Low[0], 0, Brushes.Red); |     Casting a member of the **DrawObjects[]**collection must be done safely using the "as" keyword, otherwise you may receive exceptions at run time should another instance of the object (e.g., matching tag) exist from another owner:     | ns | | --- | | NinjaScript.DrawingTools.Line myLine = DrawObjects["tag1"] as DrawingTools.Line; |     **DrawingTools** anchor fields such as "Time" or "Price", etc have been moved to a **ChartAnchor** object owned by the drawing tool, rather than a direct field on the drawing object interface.  Please refer to the NinjaTrader 8 documentation for specific changes for each drawing tool:     | ns | | --- | | double linePrice = myLine.StartAnchor.Price; |     Objects which previously used**System.Drawing.Font** now uses new **NinjaTrader.Gui.Tools.SimpleFont** class:     | ns | | --- | | Gui.Tools.SimpleFont myFont = new Gui.Tools.SimpleFont("Arial", 12); |     Properties and other methods/objects which previously [System.Drawing.Color](https://msdn.microsoft.com/en-us/library/system.drawing.color(v=vs.110).aspx) structure now use the [System.Windows.Media.Brushes](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx) class:     | ns | | --- | | BackBrush = Brushes.Blue; |      |  | | --- | | **Note**:  For custom **Brush** objects, it is important to .**Freeze()** the **Brush** due to the multi-threaded architecture of NinjaTrader 8.  Please be sure to review the new information on using [Brushes](https://ninjatrader.com/es/support/helpGuides/nt8/brushes.htm) |     **Namespaces**  The NinjaTrader 7 namespaces**NinjaTrader.Indicator** and **NinjaTrader.Strategy**have been renamed and moved to single **NinjaTrader.NinjaScript** namespace     | ns | | --- | | //This namespace holds indicators in this folder and is required. Do not change it. namespace NinjaTrader.NinjaScript.Indicators {   public class MyCustomIndicator : Indicator   {   } }   //This namespace holds Strategies in this folder and is required. Do not change it. namespace NinjaTrader.NinjaScript.Strategies {   public class MyCustomStrategy : Strategy   {   } } |     **Partial Classes (Porting methods and properties from UserDefinedMethods.cs)**  NinjaTrader 7 used a "UserDefinedMethods" class to define methods to be used across multiple NinjaScript indicators or strategies. In NinjaTrader 8, these pre-built partial classes have been removed to reduce a number of issues which could result from users sharing their UserDefinedMethods.cs files, or overwriting their existing files with copies from a new vendor. Partial classes are now best built manually and saved in the C:\Users\<user>\Documents\NinjaTrader 8\bin\Custom\AddOns folder.     |  | | --- | | **Warning**: If a partial class is saved in one of the folders used for specific NinjaScript objects other than AddOns (e.g., Indicators folder), auto-generated NinjaScript code may be appended to the end of the class by the NinjaScript Editor when compiled, which will cause a compilation error.  Saving these files in the AddOns folder will ensure they are still accessible and will not generate code which may be cause conflicts. |     You can use the template below as a starting point to create your partial class. If your partial class needs to inherit from a parent class, you can append the name of your desired parent class after the " : " to change the inheritance.     |  | | --- | | **Note**: Methods within your partial classes should be using the "public" modifier. |      | ns**Partial Class Example Template** | | --- | | namespace NinjaTrader.NinjaScript.Indicators {   public partial class MyMethods *// : parent class to inherit from*   {       //Sample method which calculates the delta of two prices       public double calculateDelta(double firstPrice, double secondPrice)       {           return Math.Abs(firstPrice - secondPrice);       }         //Sample method which prints Position information       public void printPositionInfo(Position position)       {           Print(String.Format("{0}: {1} {2} at {3}", position.Instrument, position.Quantity, position.MarketPosition, position.AveragePrice));       }           } } |     Below is an example of using one of the methods in this partial class from within an Indicator:     | ns**Partial Class Usage** | | --- | | protected override void OnBarUpdate() {   if (CurrentBar < 1) return;     // Use the static calculateDelta method to calculate the difference between the close of each bar   double delta = MyMethods.calculateDelta(Close[0], Close[1]);     Print(delta); } |      |  | | --- | | **Tip**:  At the time of the Beta implementation, the NinjaScript Editor does **NOT** include a partial class generator wizard, as it does for core NinjaScript Types such as Drawing Tools, Market Analyzer Columns, or Strategies. However, we are currently tracking a suggestion to implement a wizard for partial classes, under ID # **SFT-341**.   Please feel free to contact platformsupport@ninjatrader.com if you would like to add your vote for this enhancement. |     **Prevention of Redundant Data Loading**  In NinjaTrader 7, multiple Data Series could be added within a script, such as an indicator, and that script could then be hosted by another script, such as a strategy. While this is still possible in NinjaTrader 8, there is a new safeguard in place to prevent redundant data loading in both the hosting script and the hosted indicator.    When hosting an indicator which adds Data Series programmatically, the hosting script must include the same calls to the AddDataSeries() method as the hosted script. Without this, an error will result, which reads *"A hosted indicator tried to load additional data. All data must first be loaded by the hosting NinjaScript in its Configure state."* Without this safegaurd in place, it would be possible for unnecessarily large amounts of data to be loaded concurrently, as would be the case in a direct call to an indicator method on each OnBarUpdate(). By adding the calls to AddDataSeries() to the hosting script, you can ensure that the data is loaded when needed. Also, when this is done in the hosting script, all identical calls to AddDataSeries() in the hosted script will be ignored, as the data is already available.    The examples below show this in action:     | ns**Hosted Indicator Loads Additional Data** | | --- | | public class MyCustomIndicator : Indicator {   protected override void OnStateChange()   {     if (State == State.Configure)     {           AddDataSeries("AAPL", BarsPeriodType.Day, 1);           AddDataSeries("EURUSD", BarsPeriodType.Minute, 15);       }   } } |      | ns**Hosting Strategy Mirrors AddDataSeries() calls** | | --- | | public class MyCustomStrategy : Strategy {   // Define a MyCustomIndicator   MyCustomIndicator myIndicator;     protected override void OnStateChange()   {     if (State == State.Configure)     {         // Instantiate the MyCustomIndicator and add it to the chart         myIndicator = MyCustomIndicator();         AddChartIndicator(myIndicator);           // These calls to AddDataSeries() mirror the calls in the hosted indicator         AddDataSeries("AAPL", BarsPeriodType.Day, 1);         AddDataSeries("EURUSD", BarsPeriodType.Minute, 15);     }   } } |     **Bars with 0 Volume**  In previous versions, the NinjaTrader core was designed to replace a tick with a volume of 0 with a volume of 1.  This resulted in all ticks having a volume value of at least 1.  NinjaTrader 8 has removed that design policy and will now allow ticks with a volume of 0 to be processed.  This policy change may require logic changes to any custom bar types, indicators, or strategies which may have previously assumed volume would always be greater than 0.    **Multi-Series default "Trading Hours" templates**  The default behavior in NinjaTrader 8 will ensure that a bars series added to a script using [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) will use the same "[TradingHours](https://ninjatrader.com/es/support/helpGuides/nt8/tradinghours.htm)" template as the primary series configured by the user. In contrast, the NinjaTrader 7 behavior was highly dependent on a number of variables.  We have updated this behavior to help with consistences and synchronization issues between multiple series; however if you your script relies on two times frames using different trading hours templates, you may consider using one of the new **tradingHours**string overloaded used in [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm):     | ns | | --- | | protected override void OnStateChange() {   if (State == State.Configure)   {     // adds a 1 minute AAPL bars with a default 24/7 session tempalte.     AddDataSeries("AAPL", new BarsPeriod { BarsPeriodType = BarsPeriodType.Minute, Value = 1 }, "Default 24 x 7");   } } |     **Miscellaneous**  All of the NinjaTrader 7 reference samples posted in our support forum have been updated to demonstrate NinjaTrader 8 functionality.  Please be sure to check the reference sample section to see other undocumented features and concepts which may not have been covered in the help guide:    [Official NinjaScript reference code samples](http://www.ninjatrader.com/support/forum/forumdisplay.php?f=30)    There are several other changes to implementation which are not covered in detail on this overview, please see the code breaking changes table at the bottom of this page which will compare the implementation changes between both versions. |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?code_breaking_changes.htm#ImplementationChangesOverview)

tog_minus        [Signature Changes Overview](javascript:HMToggle('toggle','SignatureChangesOverview','SignatureChangesOverview_ICON'))

|  |  |
| --- | --- |
| **Signature**  A large number of the NinjaTrader methods which were available in NinjaTrader 7 have remained largely the same and should not generate any errors on compilation.  However there are a handful of existing methods signatures which have been updated in NinjaTrader 8 in order to fit within new framework which you would need to be aware of in order to transfer these functions from NinjaTrader 7 to NinjaTrader 8.  In most cases, the fundamental argument type has been restructured, which may result in compile errors depending on the type of object that is being used within the methods signature.     |  | | --- | | **Tip**:  Methods may now have additional signatures which add functionality which was not previously available.  Be sure to check the NinjaTrader 8 documentation which will cover all the available signatures available. | |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?code_breaking_changes.htm#SignatureChangesOverview)

tog_minus        [Name Changes Overview](javascript:HMToggle('toggle','NameChangesOverview','NameChangesOverview_ICON'))

|  |
| --- |
| **Renamed**  During the NinjaTrader 8 development process, one of our goals to make sure that our core framework matched various coding standards which have been set out in the industry.  As a result of meeting these coding standards, many NinjaTrader methods and properties needed to been renamed.    While the functionality of these methods and properties remains the same, we chose to rename these variables to follow a semantically context specific naming convention which is generally agreed upon to favor readability.  We feel that the renaming of these properties and methods more explicitly describes the intended function to the developer who may be reviewing code.  The largest number of changes is in response to the name convention of bools, where they now follow a more strict verb-adjective or verb-noun structure.    For an example:    •The property **FirstTickOfBar** may have been hard to distinguish precisely what it represented without having to look up documentation.  In NinjaTrader 8, this property has been renamed to **IsFirstTickOfBar**, which now gives this property a more readable identifier name when you read this line of code as "*is the first tick of bar true?*"  •Another example is the case of **BarsSinceEntry()** which was renamed to**BarsSinceEntryExecution()**, which now specifies that this method is looking for an entry *execution*.  •NinjaTrader 7 sometimes had methods or properties which shared names, but references different data or actions.  For example **Add()** could have been used in reference to adding **DataSeries** to a script, adding a **Plot**, or adding a **Line**.  To be more specific, NinjaTrader 8 has renamed these to **AddDataSeries()**,**AddPlot()**, and **AddLine()** respectively.  •There may be cases where the property or method name has changed simply because the type of data it interacted with has changed.  (e.g., **BarColor** vs. **BarBrush**)  •There are other cases where properties may have used unnecessary brevity and was renamed to favor readability (e.g., **AvgPrice** vs **AveragePrice**)    These are just a few examples of the many name changes found in NinjaTrader 8 and some of the rational behind the number of these changes.  For simplicity, you will find a list of all the renamed properties in the table at the bottom of this document by filtering by the "Renamed" keyword. |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?code_breaking_changes.htm#NameChangesOverview)

**Code Breaking Table**

Below you will find a reference table which lists all of the supported NinjaScript changes between NinjaTrader 7 and NinjaTrader 8.

Show 255075100All entries

Filter results:

| **Category** | **Base** | **NT7 Method/Property** | **NT8 Method/Property** | **Summary** |  |
| --- | --- | --- | --- | --- | --- |
| Implementation | Strategy | [GetAccountValue()](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?getaccountvalue.htm) | [Account.Get()](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?get.htm) | Access to Account values have been directly exposed | Details |
| Renamed | Strategy | [Add() - Strategy](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?add2.htm) | [AddChartIndicator()](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?addchartindicator.htm) | Method renamed to be more specific |  |
| Implementation | General | [Add() - Data](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?add3.htm) | [AddDataSeries()](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?adddataseries.htm) | Method was renamed to be more specific, received a number of enhancements. | Details |
| Signature | General | [AddKagi()](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?addkagi.htm) | [AddKagi()](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?addkagi.htm) | Received a number of signature changes | Details |
| Renamed | Indicator | [Add() - Line](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?add.htm) | [AddLine()](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?addline.htm) | Method renamed to be more specific |  |
| Signature | General | [AddLineBreak()](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?addlinebreak.htm) | [AddLineBreak()](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?addlinebreak.htm) | Received a number of signature changes | Details |
| Renamed | Indicator | [Add() - Plot](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?add.htm) | [AddPlot()](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?addplot.htm) | Method renamed to be more specific |  |
| Signature | General | [AddPointAndFigure()](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?addpointandfigure.htm) | [AddPointAndFigure()](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?addpointandfigure.htm) | Received a number of signature changes | Details |
| Signature | General | [AddRenko()](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?addrenko.htm) | [AddRenko()](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?addrenko.htm) | Received a number of signature changes | Details |
| Signature | General | [Alert()](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?alert.htm) | [Alert()](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?alert.htm) | Color no longer used, use Brushes instead; soundLocation now requires absolute file path | Details |
| Implementation | Drawing | [IAndrewsPitchfork](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?iandrewspitchfork.htm) | [AndrewsPitchfork](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?andrewspitchfork.htm) | IDrawingObjects have been replaced | Details |
| Implementation | Drawing | [IArc](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?iarc.htm) | [Arc](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?arc.htm) | IDrawingObjects have been replaced | Details |
| Renamed | Indicator | [LinesConfigurable](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?linesconfigurable.htm) | [AreLinesConfigurable](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?arelinesconfigurable.htm) | Property renamed to meet naming conventions |  |
| Renamed | Indicator | [PlotsConfigurable](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?plotsconfigurable.htm) | [ArePlotsConfigurable](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?areplotsconfigurable.htm) | Property renamed to meet naming conventions |  |
| Implementation | Drawing | [IArrowDown](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?iarrowdown.htm) | [ArrowDown](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?arrowdown.htm) | IDrawingObjects have been replaced | Details |
| Implementation | Drawing | [IArrowLine](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?iarrowline.htm) | [ArrowLine](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?arrowline.htm) | IDrawingObjects have been replaced | Details |
| Implementation | Drawing | [IArrowUp](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?iarrowup.htm) | [ArrowUp](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?arrowup.htm) | IDrawingObjects have been replaced | Details |
| Implementation | General | [DataSeries.Set()](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?dataseries_class.htm) | [Assignment Operator (=)](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?seriest.htm) | The .Set() method has been replaced | Details |
| Implementation | Strategy | [AtmStrategyCreate()](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?atmstrategycreate.htm) | [AtmStrategyCreate()](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?atmstrategycreate.htm) | Added a callback signature parameter | Details |
| Renamed | Strategy | [AvgBarsInTrade](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?avgbarsintrade.htm) | [AverageBarsInTrade](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?averagebarsintrade.htm) | Property renamed to favor readability |  |
| Renamed | Strategy | [AvgEtd](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?avgetd.htm) | [AverageEtd](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?averageetd.htm) | Property renamed to favor readability |  |
| Renamed | Strategy | [AvgMae](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?avgmae.htm) | [AverageMae](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?averagemae.htm) | Property renamed to favor readability |  |
| Renamed | Strategy | [AvgMfe](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?avgmfe.htm) | [AverageMfe](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?averagemfe.htm) | Property renamed to favor readability |  |
| Renamed | Strategy | [AvgPrice](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?avgprice.htm) | [AveragePrice](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?position_averageprice.htm) | Property renamed to favor readability |  |
| Renamed | Strategy | [AvgProfit](http://www.ninjatrader.com/support/helpGuides/nt7/index.html?avgprofit.htm) | [AverageProfit](http://www.ninjatrader.com/support/helpGuides/nt8/en-us/index.html?averageprofit.htm) | Property renamed to favor readability |  |

Showing 1 to 25 of 197 entries

Previous12345…8Next

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?code_breaking_changes.htm#codebreakingtable)

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [System Indicator Methods](https://ninjatrader.com/es/support/helpGuides/nt8/indicators.htm) >  **Camarilla Pivots** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/buysellvolume.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicators.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/candlestickpattern.htm) |

**Description**

Camarilla pivots are a price analysis too that generates potential support and resistance levels by multiplying the prior range then adding or subtracting it from the close.

**Syntax**

Pivots(PivotRange pivotRangeType, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow*, int *width*)  
Pivots(ISeries<double> *input*, PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*)

Returns R1 value  
Pivots(PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).R1[int *barsAgo*]  
Pivots(ISeries<double> *input*, PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).R1[int *barsAgo*]

Returns R2 value  
Pivots(PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).R2[int *barsAgo*]  
Pivots(ISeries<double> *input*, PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).R2[int *barsAgo*]

Returns R3 value  
Pivots(PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).R3[int *barsAgo*]  
Pivots(ISeries<double> *input*, PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).R3[int *barsAgo*]

Returns R4 value  
Pivots(PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).R3[int *barsAgo*]  
Pivots(ISeries<double> *input*, PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).R4[int *barsAgo*]

Returns S1 value  
Pivots(PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).S1[int *barsAgo*]  
Pivots(ISeries<double> *input*, PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).S1[int *barsAgo*]

Returns S2 value  
Pivots(PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).S2[int *barsAgo*]  
Pivots(ISeries<double> *input*, PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).S2[int *barsAgo*]

Returns S3 value  
Pivots(PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).S3[int *barsAgo*]  
Pivots(ISeries<double>*input*, PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).S3[int *barsAgo*]

Returns S4 value  
Pivots(PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).S3[int *barsAgo*]  
Pivots(ISeries<double>*input*, PivotRange *pivotRangeType*, HLCCalculationMode *priorDayHLC*, double *userDefinedClose*, double *userDefinedHigh*, double *userDefinedLow,*int *width*).S4[int *barsAgo*]

**Return Value**

double; Accessing this method via an index value [int *barsAgo*] returns the indicator value of the referenced bar.

**Parameters**

|  |  |
| --- | --- |
| input | Indicator source data ([?](https://ninjatrader.com/es/support/helpGuides/nt8/valid_input_data_for_indicator.htm)) |
| pivotRangeType | Sets the range for the type of pivot calculated. Possible values are:  PivotRange.Daily  PivotRange.Weekly  PivotRange.Monthly |
| priorDayHLC | Sets how the prior range High, Low, Close values are calculated. Possible values are:  HLCCalculationMode.CalcFromIntradayData  HLCCalculationMode.DailyBars  HLCCalculationMode.UserDefinedValues |
| userDefinedClose | Sets the close for Pivots calculations when using HLCCalculationMode.UserDefinedValues. |
| userDefinedHigh | Sets the high for Pivots calculations when using HLCCalculationMode.UserDefinedValues. |
| userDefinedLow | Sets the low for Pivots calculations when using HLCCalculationMode.UserDefinedValues. |
| width | Sets how long the Pivots lines will be drawn |

**Examples**

| ns |
| --- |
| // Prints the current R1 pivotvalue double valueR1 = CamarillaPivots(PivotRange.Daily, HLCCalculationMode.CalcFromIntradayData, 0, 0, 0, 20).R1[0]; Print("The current Camarilla Pivots' R1 value is " + valueR1.ToString());   // Prints the current S2 pivot value double valueS2 = CamarillaPivots(PivotRange.Daily, HLCCalculationMode.CalcFromIntradayData, 0, 0, 0, 20).S2[0]; Print("The current Camarilla Pivots' S2 pivot value is " + valueS2.ToString()); |

**Source Code**

You can view this indicator method source code by selecting the menu **New > NinjaScript Editor > Indicators** within the NinjaTrader Control Center window.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tip**: When using HLCCalculationMode.DailyBars it can be expected that a value of 0 is returned when the daily bars have not been loaded yet. Due to the asynchronous nature of this indicator calling daily bars you should only access the pivot values when the indicator has loaded all required Bars objects. To ensure you are accessing accurate values you can use .[IsValidDataPoint()](https://ninjatrader.com/es/support/helpGuides/nt8/isvaliddatapoint.htm) as a check:   | ns | | --- | | // Evaluates that this is a valid pivot point value if (CamarillaPivots(PivotRange.Daily, HLCCalculationMode.DailyBars, 0, 0, 0, 20).Pp.IsValidDataPoint(0)) {     // Prints the current pivot point value     double valuePp = CamarillaPivots(PivotRange.Daily, HLCCalculationMode.DailyBars, 0, 0, 0, 20).Pp[0];     Print("The current Camarilla Pivots' pivot value is " + valuePp.ToString()); } | | | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Charts](https://ninjatrader.com/es/support/helpGuides/nt8/chart.htm) > [Rendering](https://ninjatrader.com/es/support/helpGuides/nt8/rendering.htm) >  **OnRender()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/oncalculateminmax.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/rendering.htm) [Next page](https://ninjatrader.com/es/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/es/support/helpGuides/nt8/indicator.htm), [DrawingTool](https://ninjatrader.com/es/support/helpGuides/nt8/drawingtool.htm) or [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm).

|  |
| --- |
| **Notes**:  1.Thie 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/es/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm)  2.The **OnRender()**method frequently runs once the [State](https://ninjatrader.com/es/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/es/support/helpGuides/nt8/forcerefresh.htm) for more details  4.When using the [Strategy Analyzer](https://ninjatrader.com/es/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/es/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/es/support/helpGuides/nt8/getvalueat.htm), [Bars.GetOpen()](https://ninjatrader.com/es/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/es/support/helpGuides/nt8/onbarupdate.htm), etc.).  See also [OnRenderTargetChanged()](https://ninjatrader.com/es/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/es/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/es/support/helpGuides/nt8/rendertarget.htm) requires its own brushes. You must create a brushes directly in **OnRender()** or using [OnRenderTargetChanged()](https://ninjatrader.com/es/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/es/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/es/support/helpGuides/nt8/chartcontrol.htm) object (the chart's bar-related properties and x-axis) |
| chartScale | A [ChartScale](https://ninjatrader.com/es/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/es/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/es/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:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) >  **ISeries<T>** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/url.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) [Next page](https://ninjatrader.com/es/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/es/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/es/support/helpGuides/nt8/seriest.htm) | Represents a generic custom data structure for custom development |
| [PriceSeries](https://ninjatrader.com/es/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/es/support/helpGuides/nt8/timeseries.htm) | Historical time stamps structured as an ISeries<DateTime> interface (Time[0]) |
| [VolumeSeries](https://ninjatrader.com/es/support/helpGuides/nt8/volumeseries.htm) | Historical volume data structured as an ISeries<double> interface (Volume[0]) |

**Methods and Properties**

|  |  |
| --- | --- |
| [GetValueAt()](https://ninjatrader.com/es/support/helpGuides/nt8/getvalueat.htm) | Returns the underlying input value at a specified bar index value. |
| [IsValidDataPoint()](https://ninjatrader.com/es/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/es/support/helpGuides/nt8/isvaliddatapointat.htm) | Indicates if the specified input is set at a specified bar index value. |
| [Count](https://ninjatrader.com/es/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/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Charts](https://ninjatrader.com/es/support/helpGuides/nt8/chart.htm) > [Rendering](https://ninjatrader.com/es/support/helpGuides/nt8/rendering.htm) >  **OnCalculateMinMax()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/minvalue.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/rendering.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/onrender.htm) |

**Definition**

An event driven method which is called while the chart scale is being updated.  This method is used to determine the highest and lowest value that can be used for the chart scale and is only called when the chart object is set to [IsAutoScale](https://ninjatrader.com/es/support/helpGuides/nt8/isautoscale.htm).

|  |
| --- |
| **Note**:  The indexer used to look up a [Series<T>](https://ninjatrader.com/es/support/helpGuides/nt8/seriest.htm) value through barsAgo is **NOT** guaranteed to be in sync when the OnCalculateMinMax() method is called.  You will need to use [GetValueAt()](https://ninjatrader.com/es/support/helpGuides/nt8/getvalueat.htm) to obtain a historical value at a specified absolute index. |

**Method Return Value**

This method does not return a value.

**Syntax**  
You must override the method in your NinjaScript object with the following syntax:

public override void OnCalculateMinMax()  
{  
   
}

**Method Parameters**

This method does not accept any parameters.

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name       = "Example Indicator";     IsOverlay   = true;       // set this to true to ensure CalculateMinMix() is called     IsAutoScale = true;   } }   public override void OnCalculateMinMax() {   // make sure to always start fresh values to calculate new min/max values   double tmpMin = double.MaxValue;   double tmpMax = double.MinValue;     // For performance optimization, only loop through what is viewable on the chart   for (int index = ChartBars.FromIndex; index <= ChartBars.ToIndex; index++)   {     // since using Close[0] is not guaranteed to be in sync     // retrieve "Close" value at the current viewable range index     double plotValue = Close.GetValueAt(index);       // return min/max of close value     tmpMin = Math.Min(tmpMin, plotValue);     tmpMax = Math.Max(tmpMax, plotValue);   }     // Finally, set the minimum and maximum Y-Axis values to +/- 50 ticks from the primary close value   MinValue = tmpMin - 50 \* TickSize;   MaxValue = tmpMax + 50 \* TickSize; } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) >  **AddDataSeries()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/addheikenashi.htm) |

**Definition**

Adds a Bars object for developing a multi-series (multi-time frame or multi-instrument) NinjaScript.

**Related Methods and Properties**

|  |  |
| --- | --- |
| [AddHeikenAshi()](https://ninjatrader.com/es/support/helpGuides/nt8/addheikenashi.htm) | This method adds a Heiken Ashi Bars object for multi-series NinjaScript. |
| [AddKagi()](https://ninjatrader.com/es/support/helpGuides/nt8/addkagi.htm) | This method adds a Kagi Bars object for multi-series NinjaScript. |
| [AddLineBreak()](https://ninjatrader.com/es/support/helpGuides/nt8/addlinebreak.htm) | This method adds a Line Break Bars object for multi-series NinjaScript. |
| [AddPointAndFigure()](https://ninjatrader.com/es/support/helpGuides/nt8/addpointandfigure.htm) | This method adds a Point-and-Figure Bars object for multi-series NinjaScript. |
| [AddRenko()](https://ninjatrader.com/es/support/helpGuides/nt8/addrenko.htm) | This method adds a Renko Bars object for multi-series NinjaScript. |
| [AddVolumetric()](https://ninjatrader.com/es/support/helpGuides/nt8/addvolumetric.htm) | This method adds a Order Flow Volumetric Bars object for multi-series NinjaScript. |
| [BarsArray](https://ninjatrader.com/es/support/helpGuides/nt8/barsarray.htm) | An array holding Bars objects that are added via the [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) method. |
| [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) | An index value of the current Bars object that has called the [OnBarUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onbarupdate.htm) method. |
| [BarsPeriods](https://ninjatrader.com/es/support/helpGuides/nt8/barsperiods.htm) | Holds an array of BarsPeriod objects synchronized to the number of unique Bars objects held within the parent NinjaScript object. |
| [CurrentBars](https://ninjatrader.com/es/support/helpGuides/nt8/currentbars.htm) | Holds an array of int values representing the number of the current bar in a Bars object. |

**Syntax**

The following syntax will add another Bars object for the primary instrument of the script.  
AddDataSeries(BarsPeriod barsPeriod)  
AddDataSeries(BarsPeriodType periodType, int period)

The following syntax allows you to add another Bars object for a different instrument to the script:

AddDataSeries(string instrumentName, BarsPeriodType periodType, int period)  
AddDataSeries(string instrumentName, BarsPeriodType periodType, int period, MarketDataType marketDataType)  
AddDataSeries(string instrumentName, BarsPeriod barsPeriod)  
AddDataSeries(string instrumentName, BarsPeriod barsPeriod, string tradingHoursName)  
AddDataSeries(string instrumentName, BarsPeriod barsPeriod, string tradingHoursName, bool? isResetOnNewTradingDay)  
AddDataSeries(string instrumentName, BarsPeriod barsPeriod, int barsToLoad, string tradingHoursName, bool? isResetOnNewTradingDay)

AddDataSeries(string instrumentName) //only for R15 and higher

|  |
| --- |
| **Warning:**  •This method should **ONLY** be called from the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm)method during **State.Configure**  •Should your script be the host for other scripts that are creating indicators and series dependent resources in **State.DataLoaded**, please make sure that the host is doing the same **AddDataSeries()** calls as those hosted scripts would. For further reference, please also review the 2nd example below and the 'Adding additional Bars Objects to NinjaScript' section in [Multi-Time Frame & Instruments](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm)  •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/es/support/helpGuides/nt8/state.htm) (e.g., [Instrument](https://ninjatrader.com/es/support/helpGuides/nt8/instrument.htm), [Bars](https://ninjatrader.com/es/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 adding multiple Data Series of the same instrument and the same Bar Type, the 'barsToLoad' property will only be effective on the first added series. Subsequent series with a different barsToLoad setting will not load a different number of bars then the first series.  •The AddDataSeries(string instrumentName) overload allows loading a different instrument yet using the same BarsPeriod. This could not be supported for [Strategy Analyzer use with the 'Optimize Data Series'](https://ninjatrader.com/es/support/helpGuides/nt8/optimize_a_strategy.htm) option enabled, doing so 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.**  •If your NinjaScript object is using AddDataSeries() allowing to specify a tradingHoursName, please keep in mind that: 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 template of all added series. Any ticks not processed will be queued and processed as a tick comes in for all subsequent DataSeries.  •When instantiating indicators in a [Multi-Series script](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm) in [OnStateChange](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm), the input any hosted indicator is running on should be explicitly stated |

**Parameters**

|  |  |
| --- | --- |
| instrumentName | A string determining instrument name such as "MSFT" |
| barsPeriod | The [BarsPeriod](https://ninjatrader.com/es/support/helpGuides/nt8/barsperiod.htm) object (period type and interval) |
| periodType | The BarsType used for the bars period    Possible values are:    •BarsPeriodType.Tick  •BarsPeriodType.Volume  •BarsPeriodType.Range  •BarsPeriodType.Second  •BarsPeriodType.Minute  •BarsPeriodType.Day  •BarsPeriodType.Week  •BarsPeriodType.Month  •BarsPeriodType.Year |
| period | An int determining the period interval such as "3" for 3 minute bars |
| marketDataType | The MarketDataType used for the bars object (last, bid, ask)    Possible values are:    •MarketDataType.Ask  •MarketDataType.Bid  •MarketDataType.Last    **Note**: Please see the article [here](https://ninjatrader.com/es/support/helpGuides/nt8/using_historical_bid_ask_serie.htm) on using Bid/Ask series. |
| tradingHoursName | A string determining the trading hours template for the instrument |
| isResetOnNewTradingDay | A nullable bool\* determining if the Bars object should [Break at EOD](https://ninjatrader.com/es/support/helpGuides/nt8/break_at_eod.htm)    \*Will accept true, false or null as the input.  If null is used, the data series will use the settings of the primary data series. |
| barsToLoad | An int determining the number of historical bars to load |

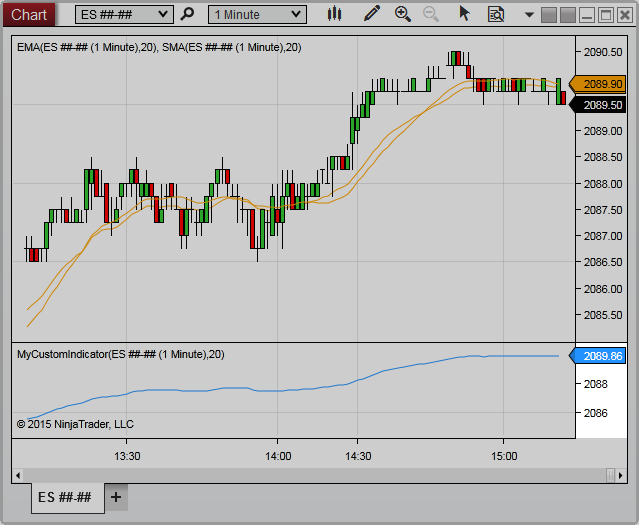
|  |
| --- |
| **Tips**:  1. You can optionally add the exchange name as a suffix to the symbol name. This is only advised if the instrument has multiple possible exchanges that it can trade on and it is configured within the Instruments window. For example: AddDataSeries("MSFT Arca", BarsPeriodType.Minute, 5);  2. You can add a custom [BarsType](https://ninjatrader.com/es/support/helpGuides/nt8/bars_type.htm) which is installed on your system by casting the registered enum value for that BarsPeriodType.  For example: AddDataSeries((BarsPeriodType)14, 10);  3. You can specify optional [BarsPeriod](https://ninjatrader.com/es/support/helpGuides/nt8/barsperiod.htm) values (such as [Value2](https://ninjatrader.com/es/support/helpGuides/nt8/optimization_fitness_value.htm)) of a custom BarsType in the BarsPeriod object initializer.  For example: AddDataSeries(new BarsPeriod() { BarsPeriodType = (BarsPeriodType)14, Value = 10, Value2 = 20 });  4. For the instrument name parameter null could be passed in, resulting in the primary data series instrument being used. |

**Examples**

| ns |
| --- |
| protected override void OnStateChange() {     if (State == State.Configure)     {         // Add a 5 minute Bars object - BarsInProgress index = 1         AddDataSeries(BarsPeriodType.Minute, 5);           // Add a 100 tick Bars object for the ES 09-16 contract - BarsInProgress index = 2         AddDataSeries("ES 09-16", BarsPeriodType.Tick, 100);     } }   protected override void OnBarUpdate() {     // Ignore bar update events for the supplementary - Bars object added above     if (BarsInProgress == 1 || BarsInProgress == 2)         return;       // Go long if we have three up bars on all bars objects     if (Close[0] > Open[0] && Closes[1][0] > Opens[1][0] && Closes[2][0] > Opens[2][0])         EnterLong(); } |

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.Configure)     {         // Our hosting script needs to have the AddDataSeries call included as well, which the Pivots indicator we call in the 2nd statement below           // also has per default in it's own State.Configure method. This is required since our Pivots indicator below is created in State.DataLoaded           // (which is happening after State.Configure and it depends on the AddDataSeries call to have the bars available to properly calculate in           // daily bars mode.         AddDataSeries(BarsPeriodType.Day, 1);     }      else if (State == State.DataLoaded)     {         //In this state, we pass the 1 day series to the Pivots indicator (as BarsArray[1]) and create its instance         pivots = Pivots(BarsArray[1], PivotRange.Weekly, HLCCalculationMode.DailyBars, 0, 0, 0, 20);     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Charts](https://ninjatrader.com/es/support/helpGuides/nt8/chart.htm) >  **ChartScale** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/y_coordinate_chartpanel.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chart.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getpixelsfordistance.htm) |

The ChartScale class includes a range of properties related to the Y-Axis values of the [ChartPanel](https://ninjatrader.com/es/support/helpGuides/nt8/chartpanel.htm) on which the calling script resides.  The ChartScale can be configured to Right, Left, or Overlay.



**Methods and Properties**

|  |  |
| --- | --- |
| [GetPixelsForDistance()](https://ninjatrader.com/es/support/helpGuides/nt8/getpixelsfordistance.htm) | Returns the number of device pixels between the value passed to the method representing a series point value on the chart scale |
| [GetValueByY()](https://ninjatrader.com/es/support/helpGuides/nt8/getvaluebyy.htm) | Returns the series value on the chart scale determined by a y pixel coordinate on the chart |
| [GetValueByYWpf()](https://ninjatrader.com/es/support/helpGuides/nt8/getvaluebyywpf.htm) | Returns the series value on the chart scale determined by a WPF coordinate on the chart |
| [GetYByValue()](https://ninjatrader.com/es/support/helpGuides/nt8/getybyvalue.htm) | Returns the chart's y-pixel coordinate on the chart determined by a series value represented on the chart scale |
| [GetYByValueWpf()](https://ninjatrader.com/es/support/helpGuides/nt8/getybyvaluewpf.htm) | Returns a WPF coordinate on the chart determined by a series value represented on the chart scale |
| [Height](https://ninjatrader.com/es/support/helpGuides/nt8/height.htm) | Indicates the overall distance (from top to bottom) of the chart scale in device pixels |
| [IsVisible](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale_isvisible.htm) | Indicates if the chart scale is viewable on the UI |
| [MaxMinusMin](https://ninjatrader.com/es/support/helpGuides/nt8/maxminusmin.htm) | The difference between the chart scale's [MaxValue](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale_maxvalue.htm) and [MinValue](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale_minvalue.htm) represented as a y value |
| [MaxValue](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale_maxvalue.htm) | The highest displayed value on the chart scale |
| [MinValue](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale_minvalue.htm) | The lowest rendered value on the chart scale |
| [PanelIndex](https://ninjatrader.com/es/support/helpGuides/nt8/panelindex.htm) | The panel on which the chart scale resides |
| [Properties](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale_properties.htm) | Represents a number of properties available to the Chart Scale which can be configured to change the appearance of the scale |
| [ScaleJustification](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale_scalejustification.htm) | Indicates the location of the chart scale relative to the chart control |
| [Width](https://ninjatrader.com/es/support/helpGuides/nt8/width.htm) | Indicates the overall distance (from left to right) of the chart scale in device pixels |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [OnBarUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onbarupdate.htm) >  **BarsPeriod** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onbarupdate.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/onbarupdate.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/calculate.htm) |

**Definition**

The primary Bars object time frame (period type and interval).

|  |
| --- |
| **Warning**:  This property should **NOT** be accessed within the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method before the **State** has reached **State.DataLoaded** |

**Property Value**

A [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) series object representing the time frame of the Bars.

**Syntax**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BarsPeriod.BarsPeriodType | The type of bars used for the period, as well as the enumeration value under which the any of the 14 default NinjaTrader types are registered. Possible values include:     |  |  | | --- | --- | | BarsPeriodType.Tick | 0 | | BarsPeriodType.Volume | 1 | | BarsPeriodType.Range | 2 | | BarsPeriodType.Second | 3 | | BarsPeriodType.Minute | 4 | | BarsPeriodType.Day | 5 | | BarsPeriodType.Week | 6 | | BarsPeriodType.Month | 7 | | BarsPeriodType.Year | 8 | | BarsPeriodType.HeikenAshi | 9 | | BarsPeriodType.Kagi | 10 | | BarsPeriodType.Renko | 11 | | BarsPeriodType.PointAndFigure | 12 | | BarsPeriodType.LineBreak | 13 | | BarsPeriodType.Volumetric | 14 |        |  | | --- | | **Tip**: When creating custom [BarsTypes](https://ninjatrader.com/es/support/helpGuides/nt8/bars_type.htm), it is recommended to pick high, unique enumeration value to avoid conflict from other BarsTypes that may be used by a single installation.    BarsPeriod = new BarsPeriod { BarsPeriodType = (BarsPeriodType)123456, BarsPeriodTypeName = "MyCustomBars", Value = 1 }; | |
| BarsPeriod.BaseBarsPeriodType | Only relevant for [HeikenAshi](https://ninjatrader.com/es/support/helpGuides/nt8/addheikenashi.htm), [Kagi](https://ninjatrader.com/es/support/helpGuides/nt8/addkagi.htm), [LineBreak](https://ninjatrader.com/es/support/helpGuides/nt8/addlinebreak.htm), [PointAndFigure](https://ninjatrader.com/es/support/helpGuides/nt8/addpointandfigure.htm) and [Volumetric](https://ninjatrader.com/es/support/helpGuides/nt8/addvolumetric.htm) Bars objects. Same possible values as BarsPeriod.BarsPeriodType |
| BarsPeriod.BaseBarsPeriodValue | Only relevant for [HeikenAshi](https://ninjatrader.com/es/support/helpGuides/nt8/addheikenashi.htm), [Kagi](https://ninjatrader.com/es/support/helpGuides/nt8/addkagi.htm), [LineBreak](https://ninjatrader.com/es/support/helpGuides/nt8/addlinebreak.htm), [PointAndFigure](https://ninjatrader.com/es/support/helpGuides/nt8/addpointandfigure.htm) and [Volumetric](https://ninjatrader.com/es/support/helpGuides/nt8/addvolumetric.htm) Bars objects. Determines an integer value representing the basePeriodTypeValue parameter |
| BarsPeriod.MarketDataType | The data type used to build the bars.  Possible values:  MarketDataType.Ask MarketDataType.Bid MarketDataType.Last |
| BarsPeriod.PointAndFigurePriceType | Only relevant for PointAndFigure Bars objects. Possible values: PointAndFigurePriceType.Close PointAndFigurePriceType.HighsAndLows |
| BarsPeriod.ReversalType | Only relevant for Kagi Bars objects. Possible values: ReversalType.Percent ReversalType.Tick |
| BarsPeriod.Value | Determines an integer value representing the period parameter.  •When using Kagi Bars objects this represents the "reversal" parameter  •When using LineBreak Bars objects this represents the "lineBreakCount" parameter  •When using PointAndFigure Bars objects this represents the "boxSize" parameter  •When using Renko Bars objects this represents the "brickSize" parameter |
| BarsPeriod.Value2 | Only relevant for PointAndFigure Bars objects. Determines an integer value representing the "reversal" parameter. |

**Examples**

| ns **Checking BarsPeriod values** |
| --- |
| // Calculate only if there is a 100 tick chart or greater protected override void OnBarUpdate() {     if (BarsPeriod.BarsPeriodType == BarsPeriodType.Tick && BarsPeriod.Value >= 100)     {         // Indicator calculation logic here     } } |

| ns **Creating a new BarsPeriod object** | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.Configure)     {                 // add a 1440 minute apple bars object using the RTH session template           AddDataSeries("AAPL", new BarsPeriod { BarsPeriodType = BarsPeriodType.Minute, Value = 1440 }, "US Equities RTH");                   }       else if (State == State.DataLoaded)     {           // Print out the loaded bars period           Print(Instrument.FullName + " " + BarsPeriod); // MSFT 1 Minute           Print(BarsArray[1].Instrument.FullName + " " + BarsArray[1].BarsPeriod); // AAPL 1440 Minute     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Add On](https://ninjatrader.com/es/support/helpGuides/nt8/add_on.htm) >  **BarsRequest** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/submit.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/add_on.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/request.htm) |

**Definition**

BarsRequest can be used to request [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) data and subscribe to real-time Bars data events.

|  |
| --- |
| **Notes**:  1.When using the DateTime fromLocal and toLocal parameters, the dates are converted to local daily timestamps (12:00 AM) and return a BarsRequest representing full trading days. If you need to request less than one full trading day, please use the barsBack parameter  2.Remember to unsubscribe from the .Update Event handler if you are no longer using the subscription.  3.A BarsRequest provides underlying market data for an instrument, but is not synchronized with an indicator or strategies primary data series.  You will need to implement your own BarsUpdateEvent logic.  4.BarsRequest data **CANNOT** be used as input for a NinjaTrader indicator  5.Performing a BarsRequest in Playback will always yield bars up to the current playback time / slider position.  6.The documented BarsRequest behavior would be the same for all NinjaScript types |

**Syntax**

BarsRequest(Cbi.Instrument instrument, int barsBack)  
BarsRequest(Cbi.Instrument instrument, DateTime fromLocal, DateTime toLocal)

**Parameters**

|  |  |
| --- | --- |
| Instrument | The [Instrument](https://ninjatrader.com/es/support/helpGuides/nt8/instrument.htm) to request |
| barsBack | An int value determining the number of bars to request from the current time |
| fromLocal | A DateTime value determining the starting date to request |
| toLocal | A DateTime value determining the ending date to request |

**Methods and Properties**

|  |  |
| --- | --- |
| Bars | The [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) object returned from the request |
| BarsBack | An int representing the number of bars back used in the request |
| BarsPeriod | The [BarsPeriod](https://ninjatrader.com/es/support/helpGuides/nt8/barsperiod.htm) for the bars request |
| FromLocal | A DateTime representing the starting date used in the request |
| IsDividendAdjusted | A bool representing if the bars request will be [dividend adjusted](https://ninjatrader.com/es/support/helpGuides/nt8/splits_and_dividends.htm) |
| IsResetOnNewTradingDay | A bool representing if the bars request will [Break at EOD](https://ninjatrader.com/es/support/helpGuides/nt8/break_at_eod.htm) |
| IsSplitAdjusted | A bool representing if the bars request will be [split adjusted](https://ninjatrader.com/es/support/helpGuides/nt8/splits_and_dividends.htm) |
| Instrument | The [Instrument](https://ninjatrader.com/es/support/helpGuides/nt8/instrument.htm) of the bars request |
| LookupPolicy | The lookup policies for the bars request.  Possible Values are:  •Provider - Queries the provider. The repository is updated on provider's reply  •Repository - Looks up the local repository only |
| [MergePolicy](https://ninjatrader.com/es/support/helpGuides/nt8/barsrequest_mergepolicy.htm) | The [merge policy](https://ninjatrader.com/es/support/helpGuides/nt8/mergepolicy.htm) for the bars request. |
| [Request()](https://ninjatrader.com/es/support/helpGuides/nt8/request.htm) | Requests the bars as parametrized |
| TradingHours | The [trading hours](https://ninjatrader.com/es/support/helpGuides/nt8/tradinghours.htm) for the bars request |
| ToLocal | A DateTime representing the end date used in the request |
| Update | A BarsUpdateEvent handler for subscribing/unsubscribing to bar update events |

**Examples**

| ns | |
| --- | --- |
| /\* Example of subscribing/unsubscribing to bars data events from an Add On as well as making bars requests. The concept can be carried over to any NinjaScript object you may be working on. \*/ public class MyAddOnTab : NTTabPage { private int daysBack = 5; private bool barsRequestSubscribed = false; private BarsRequest barsRequest;   public MyAddOnTab() {   // create a new bars request.  This will determine the insturment and range for the bars to be requested   barsRequest = new BarsRequest(Cbi.Instrument.GetInstrument("AAPL"), DateTime.Now.AddDays(-daysBack), DateTime.Now);     // Parametrize your request.   barsRequest.BarsPeriod = new BarsPeriod { BarsPeriodType = BarsPeriodType.Minute, Value = 1 };   barsRequest.TradingHours = TradingHours.Get("Default 24 x 7");     // Attach event handler for real-time events if you want to process real-time data   barsRequest.Update     += OnBarUpdate;     // Request the bars   barsRequest.Request(new Action<BarsRequest, ErrorCode, string>((bars, errorCode, errorMessage) =>   {     if (errorCode != ErrorCode.NoError)     {       // Handle any errors in requesting bars here       NinjaTrader.Code.Output.Process(string.Format("Error on requesting bars: {0}, {1}",                                       errorCode, errorMessage), PrintTo.OutputTab1);       return;     }       // Output the bars we requested. Note: The last returned bar may be a currently in-progress bar     for (int i = 0; i < bars.Bars.Count; i++)     {       // Output the bars       NinjaTrader.Code.Output.Process(string.Format("Time: {0} Open: {1} High: {2} Low: {3} Close: {4} Volume: {5}",                                       bars.Bars.GetTime(i),                                       bars.Bars.GetOpen(i),                                       bars.Bars.GetHigh(i),                                       bars.Bars.GetLow(i),                                       bars.Bars.GetClose(i),                                       bars.Bars.GetVolume(i)), PrintTo.OutputTab1);     }       // If requesting real-time bars, but there are currently no connections     lock (Connection.Connections)       if (Connection.Connections.FirstOrDefault() == null)         NinjaTrader.Code.Output.Process("Real-Time Bars: Not connected.", PrintTo.OutputTab1);   })); }   // This method is fired on real-time bar events private void OnBarUpdate(object sender, BarsUpdateEventArgs e) {   /\* Depending on the BarsPeriod type of your barsRequest you can have situations where more than one bar is    updated by a single tick. Be sure to process the full range of updated bars to ensure you did not miss a bar. \*/     // Output bar information on each tick   for (int i = e.MinIndex; i <= e.MaxIndex; i++)   {     // Processing every single tick     NinjaTrader.Code.Output.Process(string.Format("Time: {0} Open: {1} High: {2} Low: {3} Close: {4}",                                     e.BarsSeries.GetTime(i),                                     e.BarsSeries.GetOpen(i),                                     e.BarsSeries.GetHigh(i),                                     e.BarsSeries.GetLow(i),                                     e.BarsSeries.GetClose(i)), PrintTo.OutputTab1);   } }   // Called by TabControl when tab is being removed or window is closed public override void Cleanup() {   // Make sure to unsubscribe to the bars request subscription   if (barsRequest != null)     {        barsRequest.Update -= OnBarUpdate;        barsRequest.Dispose();       barsRequest = null;      } }   // Other required NTTabPage members left out for demonstration purposes. Be sure to add them in your own code. } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Add On](https://ninjatrader.com/es/support/helpGuides/nt8/add_on.htm) > [Account](https://ninjatrader.com/es/support/helpGuides/nt8/account_class.htm) >  **AccountItem** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/account_class.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/account_class.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/accountitemupdate.htm) |

**Definition**

Represents various account variables used to reflect values the status of the account.  Each account connected in NinjaTrader will have it's own unique AccountItem values.

|  |
| --- |
| **Tip**:  For strategies, see also [OnAccountItemUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onaccountitemupdate.htm).  For other objects, you can also subscribe to the [AccountItemUpdate](https://ninjatrader.com/es/support/helpGuides/nt8/accountitemupdate.htm) stream. |

**Syntax**

AccountItem

**Parameters**

|  |
| --- |
| AccountItem.BuyingPower |
| AccountItem.CashValue |
| AccountItem.Commission |
| AccountItem.ExcessIntradayMargin |
| AccountItem.ExcessInitialMargin |
| AccountItem.ExcessMaintenanceMargin |
| AccountItem.ExcessPositionMargin |
| AccountItem.Fee |
| AccountItem.GrossRealizedProfitLoss |
| AccountItem.InitialMargin |
| AccountItem.IntradayMargin |
| AccountItem.LongOptionValue |
| AccountItem.LookAheadMaintenanceMargin |
| AccountItem.LongStockValue |
| AccountItem.MaintenanceMargin |
| AccountItem.NetLiquidation |
| AccountItem.NetLiquidationByCurrency |
| AccountItem.PositionMargin |
| AccountItem.RealizedProfitLoss |
| AccountItem.ShortOptionValue |
| AccountItem.ShortStockValue |
| AccountItem.SodCashValue |
| AccountItem.SodLiquidatingValue |
| AccountItem.UnrealizedProfitLoss |
| AccountItem.TotalCashBalance |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [ISeries<T>](https://ninjatrader.com/es/support/helpGuides/nt8/iseriest.htm) >  **Series<T>** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/iseriest.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/iseriest.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/reset.htm) |

**Definition**

A Series<T> is a special generic type of data structure that can be constructed with any chosen data type and holds a series of values equal to the same number of elements as bars in a chart. If you have 200 bars loaded in your chart with a moving average plotted, the moving average itself holds a Series<double> object with 200 historical values of data, one for each bar. Series<double> objects can be used as input data for all [indicator methods](https://ninjatrader.com/es/support/helpGuides/nt8/indicators.htm). The Series<T> class implements the ISeries<T> interface.

|  |
| --- |
| **Note**:  By default NinjaTrader limits the number of values stored for Series<T> objects to 256 from the current bar being processed. This drastically improves memory performance by not holding onto old values that are generally not needed. Should you need more values than the last 256 please be sure to create the Series<T> object so that it stores all values instead through the use of the [MaximumBarsLookBack](https://ninjatrader.com/es/support/helpGuides/nt8/maximumbarslookback.htm) property. |

**Parameters**

|  |  |
| --- | --- |
| ninjaScriptBase | The NinjaScript object used to create the Series |
| bars | The [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) object used to create the Series |
| maximumBarsLookBack | A [MaximumBarsLookBack](https://ninjatrader.com/es/support/helpGuides/nt8/maximumbarslookback.htm) value used for memory performance |

**Methods and Properties**

|  |  |
| --- | --- |
| [GetValueAt()](https://ninjatrader.com/es/support/helpGuides/nt8/getvalueat.htm) | Returns the underlying input value at a specified bar index value. |
| [IsValidDataPoint()](https://ninjatrader.com/es/support/helpGuides/nt8/isvaliddatapoint.htm) | Determines if the specified input is set at a barsAgo value relative to the current bar. |
| [Reset()](https://ninjatrader.com/es/support/helpGuides/nt8/reset.htm) | Resets the internal marker which is used for [IsValidDataPoint()](https://ninjatrader.com/es/support/helpGuides/nt8/isvaliddatapoint.htm) back to false. |
| [Count](https://ninjatrader.com/es/support/helpGuides/nt8/count.htm) | The total number of bars or data points. |

**Creating Series<T> Objects**

When creating custom indicators, Series<double> objects are automatically created for you by calling the [AddPlot()](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm) method and can be subsequently referenced by the [Value](https://ninjatrader.com/es/support/helpGuides/nt8/value.htm) and/or [Values](https://ninjatrader.com/es/support/helpGuides/nt8/values.htm) property. However, you may have a requirement to create a Series<T> object to store values that are part of an overall indicator value calculation. This can be done within a custom indicator or strategy.

|  |
| --- |
| **Note**:  Custom Series<T> objects will hold the number of values specified by the [MaximumBarsLookBack](https://ninjatrader.com/es/support/helpGuides/nt8/maximumbarslookback.htm) property when the custom series object is instantiated. |

To create a Series<T> object:

1.Determine the data type of the Series<T> object you wish to create. This could be double, bool, int, string or any other object type you want.

2.Define a variable of type Series<T> that will hold a Series<T> object. This example will create "myDoubleSeries" as a Series<double>.

3.In the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method, in the State.DataLoaded create a new Series<T> object and assign it to the "myDoubleSeries" variable

| ns |
| --- |
| private Series<double> myDoubleSeries; // Define a Series<T> variable. In this instance we want it                                       // as a double so we created a Series<double> variable.   // Create a Series object and assign it to the variable protected override void OnStateChange() {     if (State == State.DataLoaded)     {         // "this" refers to the NinjaScript object itself. This syncs the Series object to historical data bars         // MaximumBarsLookBack determines how many values the Series<double> will have access to         myDoubleSeries = new Series<double>(this, MaximumBarsLookBack.Infinite);     } } |

|  |
| --- |
| **Tip***:*Series<T> objects can be used on supplementary series in a multi-time frame and instrument strategy. Please see our [support forum](http://www.ninjatrader.com/support/forum/showthread.php?t=3572) NinjaScript reference samples section for further information. |

**Setting Values**

You can set the value for the current bar being evaluated by choosing a "barsAgo" value of "0" or, for historical bars, by choosing a "barsAgo" value that represents the number of bars ago that you want the value to be stored at.

| ns **Setting Series<T> values** |
| --- |
| protected override void OnBarUpdate() {     myDoubleSeries[0] = Close[0]; } |

|  |
| --- |
| **Note**:  The "barsAgo" value is only guaranteed to be in sync with the recent current bar during core data event methods, such as OnBarUpdate(), OnMarketUpdate(), and during strategy related order events such as OnOrderUpdate(), OnExecutionUpdate(), OnPositionUpdate().  For scenarios where you may need to set a value outside of a core data/order event, such as OnRender() or a custom event, you must first synchronize the "barsAgo" pointer via the [TriggerCustomEvent()](https://ninjatrader.com/es/support/helpGuides/nt8/triggercustomevent.htm) method. |

**Checking for Valid Values**  
It is possible that you may use a Series<T> object but decide not to set a value for a specific bar. However, you should *not* try to access a Series<T>value that has not been set. Internally, a dummy value does exists, but you want to check to see if it was a valid value that you set before trying to access it for use in your calculations.  Please see [IsValidDataPoint()](https://ninjatrader.com/es/support/helpGuides/nt8/isvaliddatapoint.htm) more information.

|  |
| --- |
| **Warning**:  Calling IsValidDataPoint() will only work a [MaximumBarsLookBackInfinite](https://ninjatrader.com/es/support/helpGuides/nt8/maximumbarslookback.htm) series.  Attempting to check IsValidDataPoint() MaximumBarsLookBack256 series throw an error.  Please check the Log tab of the Control Center |

**Getting Values**  
You can access Series<T> object values using the syntax Series<T>[int *barsAgo*] where barsAgo represents the data value *n* (number of bars ago).

| ns **Accessing Series object values** |
| --- |
| protected override void OnBarUpdate() {   // Prints the current and last bar value   Print("The values are " + myDoubleSeries[0] + " " + myDoubleSeries[1]); } |

Alternatively, you can access a value at an absolute bar index using the [GetValueAt()](https://ninjatrader.com/es/support/helpGuides/nt8/getvalueat.htm) method.

|  |
| --- |
| **Note**:  In most cases, you will access the historical price series using a core data 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 data event methods, such as OnRender(), or your own custom event.  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., [GetValueAt()](https://ninjatrader.com/es/support/helpGuides/nt8/getvalueat.htm). |

**Methods that Accept ISeries<T> as Arguments**  
All [indicator methods](https://ninjatrader.com/es/support/helpGuides/nt8/indicators.htm) accept ISeries<double> objects as arguments. Carrying from the prior examples, let's print out the 10 period simple moving average of range.

| ns **Using a custom Series object as indicator input** | |
| --- | --- |
| protected override void OnBarUpdate() {   // Calculate the range of the current bar and set the value     myDoubleSeries[0] = (High[0] - Low[0]);       // Print the current 10 period SMA of range     Print("Value is " + SMA(myDoubleSeries, 10)[0]);         } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [ISeries<T>](https://ninjatrader.com/es/support/helpGuides/nt8/iseriest.htm) >  **MaximumBarsLookBack** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isvaliddatapointat.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/iseriest.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/onbarupdate.htm) |

**Definition**

Determines memory performance of custom [Series<T>](https://ninjatrader.com/es/support/helpGuides/nt8/seriest.htm) objects (such as Series<double>, Series<long>, etc.).  When using **MaximumBarsLookBack.TwoHundredFiftySix**, only the last 256 values of the series object will be stored in memory and be accessible for reference. This results in significant memory savings when using multiple series objects. In the rare case should you need older values you can use **MaximumBarsLookBack.Infinite** to allow full access of the series.

|  |
| --- |
| **Notes**:  •ISeries<T> objects that hold bar data (such as Close, High, Volume, Time, etc) always use **MaximumBarsLookBack.Infinite** which ensures all data points are always accessible during the lifetime of your NinjaScript indicator or strategy.  •Series<double> objects that hold indicator [plot values](https://ninjatrader.com/es/support/helpGuides/nt8/values.htm) always use **MaximumBarsLookBack.Infinite** which ensures that charts always display the entire indicator's calculated values. |

**Property Value**

A **MaximumBarsLookBack** enum value. Default value is **MaximumBarsLookBack.TwoHundredFiftySix**

Possible values are:

|  |  |
| --- | --- |
| MaximumBarsLookBack.TwoHundredFiftySix | Only the last 256 values of the series object will be stored in memory and accessible for reference (improves memory performance) |
| MaximumBarsLookBack.Infinite | Allow full access of the series, but you will then not be able to utilize the benefits of memory optimization |

|  |
| --- |
| **Tip**:  A **MaximumBarsLookBack.TwoHundredFiftySix** series works as a circular ring buffer, which will "loop" when the series reaches full capacity.  Specifically, once there are 256 entries in the series, new data added to the series overwrite the oldest data. |

**Syntax**

MaximumBarsLookBack

**Examples**

| ns **Setting all custom series to use the default MaximumBarsLookBack** |
| --- |
| Series<double> myDoubleSeries = null; Series<string> myStringSeries = null;   protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "Example Indicator";     // Store all series values instead of only the last 256 values     MaximumBarsLookBack = MaximumBarsLookBack.Infinite;   }   else if (State == State.DataLoaded)   {     // The custom Series<t> below are all constructed using only the NinjaScriptBase object (i.e., "this")     // therefore, the Series<T> MaximumBarsLookBack is taken from the NinjaScript's configured MaximumBarsLookBack property     myDoubleSeries = new Series<double>(this);     myStringSeries = new Series<string>(this);   } } |

| ns **Optimizing custom series to use unique MaximumBarsLookBack behavior** | |
| --- | --- |
| Series<double> myDoubleSeries = null; Series<string> myStringSeries = null;   protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "Example Indicator";   }   else if (State == State.DataLoaded)   {     // The custom Series<t> below are constructed using MaximumBarsLookBack parameter     // therefore, each Series<t> will use their uniquely specified MaximumBarsLookBack properites     myDoubleSeries = new Series<double>(this, MaximumBarsLookBack.Infinite); // stores all values     myStringSeries = new Series<string>(this, MaximumBarsLookBack.TwoHundredFiftySix); // only the last 256 values (better performance)   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Editor](https://ninjatrader.com/es/support/helpGuides/nt8/editor.htm) >  **Code Snippets** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ns_wizard.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/editor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/compile_errors.htm) |

Code Snippets can provide you with useful code templates to speed up your coding process.

tog_minus        [Understanding Code Snippet shortcuts](javascript:HMToggle('toggle','UnderstandingCodeSnippetShortcuts','UnderstandingCodeSnippetShortcuts_ICON'))

|  |
| --- |
| **You can quickly add commonly used methods and code structures via**  •Short cut characters  •Clicking on your right mouse button and selecting the menu name "**Insert Code Snippet**"  •Pressing the F2 key on your keyboard |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?code_snippets.htm#UnderstandingCodeSnippetShortcuts)

tog_minus        [How to use Code Snippet shortcuts via the keyboard](javascript:HMToggle('toggle','HowToUseCodeSnippetShortcutsViaTheKeyboard','HowToUseCodeSnippetShortcutsViaTheKeyboard_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Using the keyboard**  Enter the text in the left column and press the "Tab" key within the NinjaScript Editor.    **Current Bar Values**   |  |  | | --- | --- | | cb | CurrentBar | | o | Open[0] | | h | High[0] | | l | Low[0] | | v | Volume[0] | | i | Input[0] |       **Previous Bar Values**   |  |  | | --- | --- | | c1 | Close[1] | | o1 | Open[1] | | h1 | High[1] | | l1 | Low[1] | | v1 | Volume[1] | | i1 | Input[1] |       **Indicator Plotting**   |  |  | | --- | --- | | line | AddLine(new Stroke(Brushes.Blue, 1), 0, "Line"); | | plot | AddPlot(new Stroke(Brushes.Blue, 1), PlotStyle.Line, "Plot"); |       **Arithmetic**   |  |  | | --- | --- | | abs | Math.Abs(value) | | min | Math.Min(value1, value2) | | max | Math.Max(value1, value2) |       **Event Handler Callback Methods**   |  |  | | --- | --- | | account | protected override void OnAccountItemUpdate(Account account, AccountItem accountItem, double value) {   } | | trade | protected override void OnAddTrade(Cbi.Trade trade) {   } | | barschange | public override void OnBarsChanged() {   } | | minmax | public override void OnCalculateMinMax() {   // It is important to set MinValue and MaxValue to the min/max Y values your drawing tool uses if you want it to support auto scale } | | calcperf | protected override void OnCalculatePerformanceValue(StrategyBase strategy) {   } | | connection | protected override void OnConnectionStatusUpdate(ConnectionStatus orderStatus, ConnectionStatus priceStatus) {   } | | datapoint | protected override void OnDataPoint(Bars bars, double open, double high,               double low, double close, DateTime time,               long volume, bool isBar, double bid, double ask)               {                 } | | execution | protected override void OnExecutionUpdate(Execution execution, string executionId, double price,         int quantity, MarketPosition marketPosition, string orderId, DateTime time)         {           } | | fundamental | protected override void OnFundamentalData(FundamentalDataEventArgs fundamentalDataUpdate) {   } | | data | protected override void OnMarketData(MarketDataEventArgs marketDataUpdate) {   } | | depth | protected override void OnMarketDepth(MarketDepthEventArgs marketDepthUpdate) {   } | | mergeperf | protected override void OnMergePerformanceMetric(PerformanceMetricBase merge) {   } | | moused | public override void OnMouseDown(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {   } | | mousem | public override void OnMouseMove(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {   } | | mouseu | public override void OnMouseUp(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {   } | | optimize | protected override void OnOptimize() {   } | | ordert | protected override void OnOrderTrace(DateTime timestamp, string message) {   } | | orderu | protected override void OnOrderUpdate(Order order, double limitPrice, double stopPrice,                                 int quantity, int filled, double averageFillPrice,                                 OrderState orderState, DateTime time, ErrorCode error,                                 string nativeError)                                 {                                   } | | position | protected override void OnPositionUpdate(Position position, double averagePrice, int quantity, MarketPosition marketPosition) {   } | | render | protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   } | | windowc | protected override void OnWindowCreated(Window window) {   } | | windowd | protected override void OnWindowDestroyed(Window window) {   } |     **Control Statements**   |  |  | | --- | --- | | if | if (expression)  {    }  else  {    } | | for | for (int index = 0; index < count; index++)  {    } | | switch | switch (expression)  {    case value1:          break;    case value2:          break;     default:          break;  } |     **Drawing**     |  |  | | --- | --- | | **Shortcut** | **Method Signature** | | dap | Draw.AndrewsPitchfork(this, "MyAndrewsPitchfork", false, 10, Close[10], 5,  High[5], 0, Low[5], Brushes.Blue, DashStyleHelper.Solid, 1); | | da | Draw.Arc(this, "MyDrawArc", false, 10, Close[10], 0,  Close[0], Brushes.LimeGreen, DashStyleHelper.Dot, 2); | | dd | Draw.ArrowDown(this, "MyArrowDown", false, 0, High[0], Brushes.Red); | | du | Draw.ArrowUp(this, "MyArrowUp", false, 0, Low[0], Brushes.Red); | | ddi | Draw.Diamond(this, "MyDiamond", false, 0, High[0] + 2 \* TickSize, Brushes.Blue); | | dt | Draw.Dot(this, "MyDot", false, 0, High[0] + 2 \* TickSize, Brushes.Blue); | | de | Draw.Ellipse(this, "MyEllipse", 10, Low[10], 0, High[0], Brushes.Blue); | | di | Draw.ExtendedLine(this, "MyExtendedLine", 10, Close[10], 0, Close[0], Brushes.Blue); | | dfc | Draw.FibonacciCircle(this, "MyFibonacciCircle", true, 10, Close[10], 0, Close[0]); | | dfe | Draw.FibonacciExtensions(this, "MyFibonacciExtensions", true, 15, Close[15],  10, Close[10], 5, Close[5]); | | dfr | Draw.FibonacciRetracements(this, "MyFibonacciRetracements", false, 10, Close[10], 0, Close[0]); | | dft | Draw.FibonacciTimeExtensions(this, "MyFibonacciTimeExtensions", false, 10, Close[10], 0, Close[0]); | | dg | Draw.GannFan(this, "MyGannFan", true, 10, Close[10]); | | dh | Draw.HorizontalLine(this, "MyHorizontalLine", Close[0], Brushes.Blue); | | dl | Draw.Line(this, "MyLine", 10, Close[10], 0, Close[0], Brushes.Blue); | | dy | Draw.Ray(this, "MyRay", 10, Close[10], 0, Close[0], Brushes.Blue); | | dr | Draw.Rectangle(this, "MyRectangle", 10, Low[10], 0, High[0], Brushes.Blue); | | dre | Draw.Region(this, "MyRegion", CurrentBar, 0, Bollinger(2, 14).Upper,  Bollinger(2, 14).Lower, Brushes.Green, Brushes.Blue, 50); | | drx | Draw.RegionHighlightX(this, "MyRegionHighlightX", 10, 0, Brushes.Blue); | | dry | Draw.RegionHighlightY(this, "MyRegionHighlightY", High[0], Low[0], Brushes.Blue, Brushes.Green, 20); | | drr | Draw.RiskReward(this, "MyRiskReward", false, 0, High[0], 10, Low[0], 2, true); | | dru | Draw.Ruler(this, "tag1", true, 4, Low[4], 3, High[3], 1, Low[1]); | | ds | Draw.Square(this, "MySquare", false, 0, High[0] + 2 \* TickSize, Brushes.Blue); | | dx | Draw.Text(this, "MyText", "Sample text ", 0, High[0] + 2 \* TickSize, Brushes.Blue); | | dxf | Draw.TextFixed(this, "MyTextFixed", "Text to draw", TextPosition.TopRight); | | dtc | Draw.TrendChannel(this, "TrendChannel", true, 10, Low[10], 0, High[0], 10, High[10] + 5 \* TickSize); | | dtd | Draw.TriangleDown(this, "MyTriangleDown", false, 0, High[0] + 2 \* TickSize, Brushes.Red); | | dtu | Draw.TriangleUp(this, "MyTriangleUp", false, 0, Low[0] - 2 \* TickSize, Brushes.Blue); | | dv | Draw.VerticalLine(this, "MyVerticalLine", 0, Brushes.Blue); | |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?code_snippets.htm#HowToUseCodeSnippetShortcutsViaTheKeyboard)

tog_minus        [How to insert Code Snippets via the mouse or F2 key](javascript:HMToggle('toggle','HowToInsertCodeSnippetsViaTheMouseOrF2Key','HowToInsertCodeSnippetsViaTheMouseOrF2Key_ICON'))

|  |
| --- |
| **Via mouse or pressing the F2 key**  1. Right mouse click in the NinjaScript Editor and select the menu name "**Insert Code Snippet**"    NS_Editor_10    2. A menu will display all available code snippets.    NS_Editor_11 |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?code_snippets.htm#HowToInsertCodeSnippetsViaTheMouseOrF2Key)

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Educational Resources](https://ninjatrader.com/es/support/helpGuides/nt8/educational_resources.htm) > [Tips](https://ninjatrader.com/es/support/helpGuides/nt8/tips.htm) >  **Using [] brackets** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/user_definable_color_inputs.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tips.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) |

In C#, square brackets represent a way to access values stored within an collection. NinjaScript comes with quite a few collections that we call ISeries objects which could be accessed with square brackets. [ISeries objects](https://ninjatrader.com/es/support/helpGuides/nt8/iseriest.htm) are linked to the underlying bars series in that they hold the same number of values as the number of bars on a chart. For example, to get the close price one bar ago, you would use Close[1] since the value of 1 within the square brackets represents the number of bars ago whose value you wish to reference. As another example, to get the high three bars ago, you would use High[3].

| ns |
| --- |
| double close1 = Close[1]; // gets the close price one bar ago  double high3 = High[3]; // gets the high of three bars ago  double low = Low; // results in compile error. Low is an array, and can't be accessed directly. It should be Low[n Bars ago]. |

Many of NinjaTrader's indicators store their values in Series as well, generally in a Plot. Plots are essentially a Series<double> object and to retrieve values from them you need to specify which value you want to access. In most cases, you'd like the current value, so you could use SMA(14)[0], not just SMA(14). SMA(14) is the Indicator its self or Series, and you can't access its values by calling it directly. Using SMA(14)[0] retrieves the part of the Series you're interested in--the most current value.

| ns |
| --- |
| double SMA\_current = SMA(14)[0]; // gets the current value of the SMA  double SMA\_1 = SMA(14)[1]; // gets the SMA value one bar ago  double SMA\_value = SMA(14); // results in compile error. SMA(14) is a Series and the variable SMA\_value of type double can't hold a Series. |

Most of the time, you need an index value (number in the square brackets), but there are also cases when you need to use the ISeries instead. CrossAbove() and CrossBelow() are two key examples. If you look at the reference page for CrossAbove(), the two method signatures (overloads) look like this:

| ns |
| --- |
| CrossAbove(ISeries<double> series1, ISeries<double> series2, int lookBackPeriod)  CrossAbove(ISeries<double> series1, double value, int lookBackPeriod) |

This means the first variable must always be a ISeries<double> object, and the second variable can be either another ISeries<double> or a double value (100, 70.25, etc). To specify a ISeries<double> object, you can just leave off the square brackets. For example if(CrossAbove(SMA(14), SMA(28), 1)) checks if the 14 period SMA has crossed above the 28 period SMA within the last bar. if(CrossAbove(SMA(14)[0], SMA(28)[0], 1)) would give you a compile error because it expects a ISeries<double> as input, not a double value (which is returned when an index is present).

| ns | |
| --- | --- |
| if (CrossAbove(SMA(14), SMA(28), 1)) // works fine  if (CrossAbove(SMA(14), 1000, 1)) // works fine, this uses a double for the second argument. See the above overload.  if (CrossAbove(SMA(14)[0], SMA(28)[0], 1)) // compile error: SMA(14)[0] is a double, not a ISeries<double>  if (CrossAbove(SMA(14), SMA(28)[0], 1)) // would work fine with a ISeries<double> as first argument and a double as the second argument | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) >  **AddKagi()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/addheikenashi.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/addlinebreak.htm) |

**Definition**

Similar to the [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) method for adding Bars objects, this method adds a Kagi Bars object for multi-series NinjaScript.

|  |
| --- |
| **Notes**:  1.When running NinjaScript, you will be able to choose the first instrument and bar interval to run on. This first Bars object will carry a [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) index of 0.  2. In a multi-time frame and multi-instrument NinjaScript, supplementary Bars objects are added via this method in State.Configure state of the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method and given an incremented BarsInProgress index value. See additional information on running [multi-bars scripts](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm).  3.The [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) property can be used to filter updates between different bars series  4.If using [OnMarketData()](https://ninjatrader.com/es/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/es/support/helpGuides/nt8/developing_for__tick_replay.htm) historical  5.For adding regular Bars types please use [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm)  6.A**Tick Replay** indicator or strategy **CANNOT**use a **MarketDataType.Ask** or **MarketDataType.Bid**series.  Please see [Developing for Tick Replay](https://ninjatrader.com/es/support/helpGuides/nt8/developing_for__tick_replay.htm) for more information. |

**Syntax**

AddKagi(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, int reversal, Data.ReversalType reversalType, Data.MarketDataType marketDataType)  
AddKagi(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, int reversal, Data.ReversalType reversalType, Data.MarketDataType marketDataType, string tradingHoursName)  
AddKagi(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, int reversal, Data.ReversalType reversalType, Data.MarketDataType marketDataType, string tradingHoursName, bool? isResetOnNewTradingDay)

|  |
| --- |
| **Warnings:**  •This method should **ONLY** be called from the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm)method during **State.Configure**  •Should your script be the host for other scripts that are creating indicators and series dependent resources in **State.DataLoaded**, please make sure that the host is doing the same **AddKagi()** calls as those hosted scripts would. For further reference, please also review the 'Adding additional Bars Objects to NinjaScript' section in [Multi-Time Frame & Instruments](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm)  **•**Arguments supplied to **AddKagi()** should be hardcoded and **NOT** dependent on run-time variables which cannot be reliably obtained during [State.Configure](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm) (e.g., [Instrument](https://ninjatrader.com/es/support/helpGuides/nt8/instrument.htm), [Bars](https://ninjatrader.com/es/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.** |

**Parameters**

|  |  |
| --- | --- |
| instrumentName | A string determining instrument name such as "MSFT" |
| baseBarsPeriodType | The underlying BarsType used for the Kagi bars period    Possible values are:    •BarsPeriodType.Day  •BarsPeriodType.Minute  •BarsPeriodType.Second  •BarsPeriodType.Tick  •BarsPeriodType.Volume |
| baseBarsPeriodTypeValue | An int determining the underlying period interval such as "3" for 3 minute bars |
| reversal | An int determining the required price movement in the reversal direction before a reversal is identified on the chart |
| reversalType | An enum determining the mode reversal period is based.    Possible values are:    •ReversalType.Percent  •ReversalType.Tick |
| marketDataType | The MarketDataType used for the bars object (last, bid, ask)    Possible values are:    •MarketDataType.Ask  •MarketDataType.Bid  •MarketDataType.Last    **Note**: Please see the article [here](https://ninjatrader.com/es/support/helpGuides/nt8/using_historical_bid_ask_serie.htm) on using Bid/Ask series. |
| tradingHoursName | A string determining the trading hours template for the instrument |
| isResetOnNewTradingDay | A nullable bool\* determining if the Bars object should [Break at EOD](https://ninjatrader.com/es/support/helpGuides/nt8/break_at_eod.htm)    \*Will accept true, false or null as the input.  If null is used, the data series will use the settings of the primary data series. |

|  |
| --- |
| **Tip**: You can optionally add the exchange name as a suffix to the symbol name. This is only advised if the instrument has multiple possible exchanges that it can trade on and it is configured within the Instruments window. For example: AddKagi("MSFT Arca", PeriodType.Minute, 1, 2, ReversalType.Tick, MarketDataType.Last) |

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {       Name = "Examples Indicator";               }   else if (State == State.Configure)   {       // Add a 1 minute Kagi Bars object for the ES 03-18 contract - BarsInProgress index = 1       AddKagi("ES 03-18", BPeriodType.Minute, 1, 2, ReversalType.Tick, MarketDataType.Last);   } }   protected override void OnBarUpdate() {     // Ignore the primary Bars object and only process the Kagi Bars object     if (BarsInProgress == 1)     {         // Do something;     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) >  **AddPointAndFigure()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/addlinebreak.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/addrenko.htm) |

**Definition**

Similar to the [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) method for adding Bars objects, this method adds a Point-and-Figure Bars object for multi-series NinjaScript.

|  |
| --- |
| **Notes**:  1.When running NinjaScript, you will be able to choose the first instrument and bar interval to run on. This first Bars object will carry a [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) index of 0.  2. In a multi-time frame and multi-instrument NinjaScript, supplementary Bars objects are added via this method in State.Configure state of the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method and given an incremented BarsInProgress index value. See additional information on running [multi-bars scripts](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm).  3.The [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) property can be used to filter updates between different bars series  4.If using [OnMarketData()](https://ninjatrader.com/es/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/es/support/helpGuides/nt8/developing_for__tick_replay.htm) historical  5.For adding regular Bars types please use [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm)  6.A**Tick Replay** indicator or strategy **CANNOT**use a **MarketDataType.Ask** or **MarketDataType.Bid**series.  Please see [Developing for Tick Replay](https://ninjatrader.com/es/support/helpGuides/nt8/developing_for__tick_replay.htm) for more information. |

**Syntax**

AddPointAndFigure(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, int boxSize, int reversal, Data.PointAndFigurePriceType pointAndFigurePriceType, Data.MarketDataType marketDataType)  
AddPointAndFigure(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, int boxSize, int reversal, Data.PointAndFigurePriceType pointAndFigurePriceType, Data.MarketDataType marketDataType, string tradingHoursName)  
AddPointAndFigure(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, int boxSize, int reversal, Data.PointAndFigurePriceType pointAndFigurePriceType, Data.MarketDataType marketDataType, string tradingHoursName, bool? isResetOnNewTradingDay)

|  |
| --- |
| **Warnings:**  •This method should **ONLY** be called from the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm)method during **State.Configure**  •Should your script be the host for other scripts that are creating indicators and series dependent resources in **State.DataLoaded**, please make sure that the host is doing the same **AddPointAndFigure()** calls as those hosted scripts would. For further reference, please also review the 'Adding additional Bars Objects to NinjaScript' section in [Multi-Time Frame & Instruments](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm)  **•**Arguments supplied to **AddPointAndFigure()** should be hardcoded and **NOT** dependent on run-time variables which cannot be reliably obtained during [State.Configure](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm) (e.g., [Instrument](https://ninjatrader.com/es/support/helpGuides/nt8/instrument.htm), [Bars](https://ninjatrader.com/es/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.** |

**Parameters**

|  |  |
| --- | --- |
| instrumentName | A string determining instrument name such as "MSFT" |
| baseBarsPeriodType | The underlying BarsType used for the Point-and-Figure bars period    Possible values are:    •BarsPeriodType.Day  •BarsPeriodType.Minute  •BarsPeriodType.Second  •BarsPeriodType.Tick  •BarsPeriodType.Volume |
| baseBarsPeriodTypeValue | An int determining the underlying period interval such as "3" for 3 minute bars |
| boxSize | An int determining the price movement signified by the X's and O's of a Point-and-Figure chart |
| reversal | An int determining the number of boxes the price needs to move in the reversal direction before a new column will be built |
| pointAndFigurePriceType | Determines where to base reversal calculations    Possible values are:    •PointAndFigurePriceType.Close  •PointAndFigurePriceType.HighsAndLows |
| marketDataType | The MarketDataType used for the bars object (last, bid, ask)    Possible values are:    •MarketDataType.Ask  •MarketDataType.Bid  •MarketDataType.Last    **Note**: Please see the article [here](https://ninjatrader.com/es/support/helpGuides/nt8/using_historical_bid_ask_serie.htm) on using Bid/Ask series. |
| tradingHoursName | A string determining the trading hours template for the instrument |
| isResetOnNewTradingDay | A nullable bool\* determining if the Bars object should [Break at EOD](https://ninjatrader.com/es/support/helpGuides/nt8/break_at_eod.htm)    \*Will accept true, false or null as the input.  If null is used, the data series will use the settings of the primary data series. |

|  |
| --- |
| **Tip**: You can optionally add the exchange name as a suffix to the symbol name. This is only advised if the instrument has multiple possible exchanges that it can trade on and it is configured within the Instruments window. For example: AddPointAndFigure("MSFT Arca", BarsPeriodType.Minute, 1, 2, 3, PointAndFigurePriceType.Close, MarketDataType.Last) |

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.Configure)     {         // Add a 1 minute Point-and-Figure Bars object for the ES 03-18 contract - BarsInProgress index = 1         AddPointAndFigure("ES 03-18", BarsPeriodType.Minute, 1, 2, 3, PointAndFigurePriceType.Close, MarketDataType.Last);     } }   protected override void OnBarUpdate() {     // Ignore the primary Bars object and only process the Point-and-Figure Bars object     if (BarsInProgress == 1)     {         // Do something;     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Charts](https://ninjatrader.com/es/support/helpGuides/nt8/chart.htm) > [ChartScale](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale.htm) >  **GetValueByY()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getpixelsfordistance.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getvaluebyywpf.htm) |

**Definition**

Returns the series value on the chart scale determined by a y pixel coordinate on the chart.

**Method Return Value**

A double value representing a series value on the chart scale.  This is normally a price value, but can represent indicator plot values as well.

**Syntax**  
<chartScale>.GetValueByY(float y)

**Method Parameters**

|  |  |
| --- | --- |
| y | A float value representing a pixel coordinate on the chart scale |

**Examples**

| ns |
| --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // the price value of the pixel coordinate passed in the method   double valueByY =   chartScale.GetValueByY(1);     Print("valueByY: " + valueByY); //2106.19693333   } |

In the image below, we pass a value of 1 for the y value, which tells us the pixel coordinate of 1 is located at a price of 2106.19 on the chart scale



|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [System Indicator Methods](https://ninjatrader.com/es/support/helpGuides/nt8/indicators.htm) >  **Correlation** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/commodity_channel_index_cci.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicators.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/current_day_ohl.htm) |

**Description**

The correlation indicator will plot the correlation of the data series to a desired instrument. Values close to 1 indicate movement in the same direction. Values close to -1 indicate movement in opposite directions. Values near 0 indicate no correlation.

**Syntax**

Correlation(int *period,*string *correlationSeries*)  
string *correlationSeies*(ISeries<double> *input*, int *period,*string *correlationSeies*)

**Return Value**

double; Accessing this method via an index value [int *barsAgo*] returns the indicator value of the referenced bar.

**Parameters**

|  |  |
| --- | --- |
| input | Indicator source data ([?](https://ninjatrader.com/es/support/helpGuides/nt8/valid_input_data_for_indicator.htm)) |
| period | Number of bars used in the calculation |
| correlationSeries | The data series to compare to |

**Examples**

| ns |
| --- |
| // The correlation data series must be added to OnStateChange() as this indicator runs off the correlation data series data else if (State == State.Configure) {   AddDataSeries("SPY"); }    // Checks the bars in progress and prints the current correlation to the SPY if (BarsInProgress == 0) {    double value = Correlation(20, "SPY")[0];   Print("The current correlation to the SPY is " + value.ToString());  } |

|  |
| --- |
| **Note**: If the correlation series does not plot during a time the input series plots, a value of zero would plot in the above example. You may consider ignroing zero values. |

**Source Code**

You can view this indicator method source code by selecting the menu **New > NinjaScript Editor > Indicators** within the NinjaTrader Control Center window.

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) >  **AddVolumetric()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/addrenko.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/barsarray.htm) |

**Definition**

Similar to the [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) method for adding Bars objects, this method adds a [Order Flow](https://ninjatrader.com/es/support/helpGuides/nt8/order_flow_volumetric_bars.htm) Volumetric Bars object for multi-series NinjaScript.

|  |
| --- |
| **Notes**:  1.When running NinjaScript, you will be able to choose the first instrument and bar interval to run on. This first Bars object will carry a [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) index of 0.  2. In a multi-time frame and multi-instrument NinjaScript, supplementary Bars objects are added via this method in State.Configure state of the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method and given an incremented BarsInProgress index value. See additional information on running [multi-bars scripts](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm).  3.The [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) property can be used to filter updates between different bars series  4.If using [OnMarketData()](https://ninjatrader.com/es/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/es/support/helpGuides/nt8/developing_for__tick_replay.htm) historical  5.For adding regular Bars types please use [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm)  6.A**Tick Replay** indicator or strategy **CANNOT**use a **MarketDataType.Ask** or **MarketDataType.Bid**series.  Please see [Developing for Tick Replay](https://ninjatrader.com/es/support/helpGuides/nt8/developing_for__tick_replay.htm) for more information.  7.To access additional Volumetric data points programmtically in your NinjaScript studies, please see the example [here](https://ninjatrader.com/es/support/helpGuides/nt8/order_flow_volumetric_bars2.htm). |

**Syntax**

AddVolumetric(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, Data.VolumetricDeltaType deltaType, int tickPerLevel)  
AddVolumetric(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, Data.VolumetricDeltaType deltaType, int tickPerLevel, bool? isResetOnNewTradingDay)  
AddVolumetric(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, Data.VolumetricDeltaType deltaType, int tickPerLevel, string tradingHoursName, bool? isResetOnNewTradingDay)

AddVolumetric(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, Data.VolumetricDeltaType deltaType, int tickPerLevel, int sizeFilter, string tradingHoursName, bool? isResetOnNewTradingDay) (R17 and higher only)

|  |
| --- |
| **Warnings:**  •This method should **ONLY** be called from the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm)method during **State.Configure**  •Should your script be the host for other scripts that are creating indicators and series dependent resources in **State.DataLoaded**, please make sure that the host is doing the same **AddVolumetric()** calls as those hosted scripts would. For further reference, please also review the 'Adding additional Bars Objects to NinjaScript' section in [Multi-Time Frame & Instruments](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm)  **•**Arguments supplied to **AddVolumetric()** should be hardcoded and **NOT** dependent on run-time variables which cannot be reliably obtained during [State.Configure](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm) (e.g., [Instrument](https://ninjatrader.com/es/support/helpGuides/nt8/instrument.htm), [Bars](https://ninjatrader.com/es/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.** |

**Parameters**

|  |  |
| --- | --- |
| instrumentName | A string determining instrument name such as "MSFT" |
| baseBarsPeriodType | The underlying BarsType used for the Volumetric bars period.    Possible values are:    •BarsPeriodType.Tick  •BarsPeriodType.Volume  •BarsPeriodType.Range  •BarsPeriodType.Second  •BarsPeriodType.Minute  •BarsPeriodType.Day  •BarsPeriodType.Week  •BarsPeriodType.Month  •BarsPeriodType.Year |
| baseBarsPeriodTypeValue | An int determining the underlying period interval such as "3" for 3 minute bars |
| deltaType | The DeltaType used for the Volumetric bars object delta calculations    Possible values are:    •VolumetricDeltaType.BidAsk  •VolumetricDetlaType.UpDownTick |
| ticksPerLevel | An int setting the aggregation of price levels for the Volumetric bar, pass in a 1 to analyze each price level individually |
| sizeFilter | An int setting the trade size allowed to count in the delta calculations |
| tradingHoursName | A string determining the trading hours template for the instrument |
| isResetOnNewTradingDay | A nullable bool\* determining if the Bars object should [Break at EOD](https://ninjatrader.com/es/support/helpGuides/nt8/break_at_eod.htm)    \*Will accept true, false or null as the input.  If null is used, the data series will use the settings of the primary data series. |

|  |
| --- |
| **Tip**: You can optionally add the exchange name as a suffix to the symbol name. This is only advised if the instrument has multiple possible exchanges that it can trade on and it is configured within the Instruments window. For example: AddVolumetric("MSFT Arca", BarsPeriodType.Minute, 1, VolumetricDeltaType.BidAsk, 1); |

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {       Name = "Examples Indicator";               }   else if (State == State.Configure)   {       // Add a 1 minute Order Flow Volumetric Bars object for the ES 03-18 contract - BarsInProgress index = 1       AddVolumetric("ES 03-18", BarsPeriodType.Minute, 1, VolumetricDeltaType.BidAsk, 1);   } }   protected override void OnBarUpdate() {     // Ignore the primary Bars object and only process the Order Flow Volumetric object     if (BarsInProgress == 1)     {         // Do something;     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) >  **AddHeikenAshi()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/addkagi.htm) |

**Definition**

Similar to the [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) method for adding Bars objects, this method adds a Heiken Ashi Bars object for multi-series NinjaScript.

|  |
| --- |
| **Notes**:  1.When running NinjaScript, you will be able to choose the first instrument and bar interval to run on. This first Bars object will carry a [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) index of 0.  2. In a multi-time frame and multi-instrument NinjaScript, supplementary Bars objects are added via this method in State.Configure state of the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method and given an incremented BarsInProgress index value. See additional information on running [multi-bars scripts](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm).  3.The [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) property can be used to filter updates between different bars series  4.If using [OnMarketData()](https://ninjatrader.com/es/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/es/support/helpGuides/nt8/developing_for__tick_replay.htm) historical  5.For adding regular Bars types please use [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm)  6.A**Tick Replay** indicator or strategy **CANNOT**use a **MarketDataType.Ask** or **MarketDataType.Bid**series.  Please see [Developing for Tick Replay](https://ninjatrader.com/es/support/helpGuides/nt8/developing_for__tick_replay.htm) for more information. |

**Syntax**

AddHeikenAshi(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, Data.MarketDataType marketDataType)  
AddHeikenAshi(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, Data.MarketDataType marketDataType, string tradingHoursName)  
AddHeikenAshi(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, Data.MarketDataType marketDataType, string tradingHoursName, bool? isResetOnNewTradingDay)

|  |
| --- |
| **Warnings:**  •This method should **ONLY** be called from the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm)method during **State.Configure**  •Should your script be the host for other scripts that are creating indicators and series dependent resources in **State.DataLoaded**, please make sure that the host is doing the same **AddHeikenAshi()** calls as those hosted scripts would. For further reference, please also review the 'Adding additional Bars Objects to NinjaScript' section in [Multi-Time Frame & Instruments](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm)  **•**Arguments supplied to **AddHeikenAshi()** should be hardcoded and **NOT** dependent on run-time variables which cannot be reliably obtained during [State.Configure](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm) (e.g., [Instrument](https://ninjatrader.com/es/support/helpGuides/nt8/instrument.htm), [Bars](https://ninjatrader.com/es/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.** |

**Parameters**

|  |  |
| --- | --- |
| instrumentName | A string determining instrument name such as "MSFT" |
| baseBarsPeriodType | The underlying BarsType used for the Heiken Ashi bars period.    Possible values are:    •BarsPeriodType.Tick  •BarsPeriodType.Volume  •BarsPeriodType.Range  •BarsPeriodType.Second  •BarsPeriodType.Minute  •BarsPeriodType.Day  •BarsPeriodType.Week  •BarsPeriodType.Month  •BarsPeriodType.Year |
| baseBarsPeriodTypeValue | An int determining the underlying period interval such as "3" for 3 minute bars |
| marketDataType | The MarketDataType used for the bars object (last, bid, ask)    Possible values are:    •MarketDataType.Ask  •MarketDataType.Bid  •MarketDataType.Last    **Note**: Please see the article [here](https://ninjatrader.com/es/support/helpGuides/nt8/using_historical_bid_ask_serie.htm) on using Bid/Ask series. |
| tradingHoursName | A string determining the trading hours template for the instrument |
| isResetOnNewTradingDay | A nullable bool\* determining if the Bars object should [Break at EOD](https://ninjatrader.com/es/support/helpGuides/nt8/break_at_eod.htm)    \*Will accept true, false or null as the input.  If null is used, the data series will use the settings of the primary data series. |

|  |
| --- |
| **Tip**: You can optionally add the exchange name as a suffix to the symbol name. This is only advised if the instrument has multiple possible exchanges that it can trade on and it is configured within the Instruments window. For example: AddHeikenAshi("MSFT Arca", BarsPeriodType.Minute, 1, MarketDataType.Last); |

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {       Name = "Examples Indicator";               }   else if (State == State.Configure)   {       // Add a 1 minute Heiken Ashi Bars object for the ES 03-18 contract - BarsInProgress index = 1       AddHeikenAshi("ES 03-18", BarsPeriodType.Minute, 1, MarketDataType.Last);   } }   protected override void OnBarUpdate() {     // Ignore the primary Bars object and only process the Heiken Ashi object     if (BarsInProgress == 1)     {         // Do something;     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) >  **AddLineBreak()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/addkagi.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/addpointandfigure.htm) |

**Definition**

Similar to the [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) method for adding Bars objects, this method adds a Line Break Bars object for multi-series NinjaScript.

|  |
| --- |
| **Notes**:  1.When running NinjaScript, you will be able to choose the first instrument and bar interval to run on. This first Bars object will carry a [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) index of 0.  2. In a multi-time frame and multi-instrument NinjaScript, supplementary Bars objects are added via this method in State.Configure state of the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method and given an incremented BarsInProgress index value. See additional information on running [multi-bars scripts](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm).  3.The [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) property can be used to filter updates between different bars series  4.If using [OnMarketData()](https://ninjatrader.com/es/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/es/support/helpGuides/nt8/developing_for__tick_replay.htm) historical  5.For adding regular Bars types please use [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm)  6.A**Tick Replay** indicator or strategy **CANNOT**use a **MarketDataType.Ask** or **MarketDataType.Bid**series.  Please see [Developing for Tick Replay](https://ninjatrader.com/es/support/helpGuides/nt8/developing_for__tick_replay.htm) for more information. |

**Syntax**

AddLineBreak(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, int lineBreakCount, Data.MarketDataType marketDataType)  
AddLineBreak(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, int lineBreakCount, Data.MarketDataType marketDataType, string tradingHoursName)  
AddLineBreak(string instrumentName, Data.BarsPeriodType baseBarsPeriodType, int baseBarsPeriodTypeValue, int lineBreakCount, Data.MarketDataType marketDataType, string tradingHoursName, bool? isResetOnNewTradingDay)

|  |
| --- |
| **Warnings:**  •This method should **ONLY** be called from the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm)method during **State.Configure**  •Should your script be the host for other scripts that are creating indicators and series dependent resources in **State.DataLoaded**, please make sure that the host is doing the same **AddLineBreak()** calls as those hosted scripts would. For further reference, please also review the 'Adding additional Bars Objects to NinjaScript' section in [Multi-Time Frame & Instruments](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm)  **•**Arguments supplied to **AddLineBreak()** should be hardcoded and **NOT** dependent on run-time variables which cannot be reliably obtained during [State.Configure](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm) (e.g., [Instrument](https://ninjatrader.com/es/support/helpGuides/nt8/instrument.htm), [Bars](https://ninjatrader.com/es/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.** |

**Parameters**

|  |  |
| --- | --- |
| instrumentName | A string determining instrument name such as "MSFT" |
| baseBarsPeriodType | The underlying BarsType used for the LineBreak bars period    Possible values are:  BarsPeriodType.Day BarsPeriodType.Minute BarsPeriodType.Second BarsPeriodType.Tick BarsPeriodType.Volume |
| baseBarsPeriodTypeValue | An int determining the underlying period interval such as "3" for 3 minute bars |
| lineBreakCount | An int determining the number of bars back used to calculate a line break |
| marketDataType | The MarketDataType used for the bars object (last, bid, ask)    Possible values are:    •MarketDataType.Ask  •MarketDataType.Bid  •MarketDataType.Last    **Note**: Please see the article [here](https://ninjatrader.com/es/support/helpGuides/nt8/using_historical_bid_ask_serie.htm) on using Bid/Ask series. |
| tradingHoursName | A string determining the trading hours template for the instrument |
| isResetOnNewTradingDay | A nullable bool\* determining if the Bars object should [Break at EOD](https://ninjatrader.com/es/support/helpGuides/nt8/break_at_eod.htm)    \*Will accept true, false or null as the input.  If null is used, the data series will use the settings of the primary data series. |

|  |
| --- |
| **Tip**: You can optionally add the exchange name as a suffix to the symbol name. This is only advised if the instrument has multiple possible exchanges that it can trade on and it is configured within the Instruments window. For example: AddLineBreak("MSFT Arca", PeriodType.Minute, 1, 3, MarketDataType.Last) |

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {         if (State == State.SetDefaults)   {     Name = "Examples Indicator";               }     if (State == State.Configure)   {     // Add a 1 minute Line Break Bars object for the ES 03-18 - BarsInProgress index = 1     AddLineBreak("ES 03-18", BarsPeriodType.Minute, 1, 3, MarketDataType.Last);   }         }   protected override void OnBarUpdate() {     // Ignore the primary Bars object and only process the Line Break Bars object     if (BarsInProgress == 1)     {         // Do something;     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Charts](https://ninjatrader.com/es/support/helpGuides/nt8/chart.htm) > [ChartScale](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale.htm) >  **GetValueByYWpf()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getvaluebyy.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getybyvalue.htm) |

**Definition**

Returns the series value on the chart scale determined by a WPF coordinate on the chart.

**Method Return Value**

A double value representing a series value on the chart scale.  This is normally a price value, but can represent indicator plot values as well.

**Syntax**  
<chartScale>.GetValueByYWpf(double y)

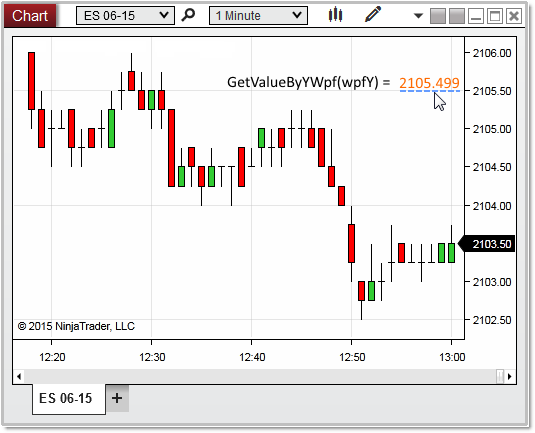
**Method Parameters**

|  |  |
| --- | --- |
| y | A double value representing a WPF coordinate on the chart scale |

**Examples**

| ns |
| --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // store the y location the user clicked   double   wpfY = chartControl.MouseDownPoint.Y;     // gets price value of the WPF coordinate passed to the method   double   valueByYWpf = chartScale.GetValueByYWpf(wpfY);     Print("valueByYWpf: " + valueByYWpf); //2105.49995215 } |

In the image below, we used the Chart Control property [MouseDownPoint](https://ninjatrader.com/es/support/helpGuides/nt8/mousedownpoint.htm) as the "wpfy" variable, which in return tells us the user clicked on a Y value of 2105.499 on the chart scale.



|  |  |
| --- | --- |
| **Navigation:**  »No topics above this level«  **8.0.5.2** | [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout.htm) |

**8.0.5.0 Release**

March 6, 2017

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 11142 | Fixed | ATM Strategies | Reversing an active ATM that had a flat position would not cancel the original ATM |
| 11299 | Fixed | Chart | In some scenarios reloading scripts rendered global draw objects invisible |
| 11284 | Fixed | Chart | Resolved DirectX errors that occurred in some scenarios |
| 11260 | Fixed | Chart | Muli-series charts were plotting future time axis values incorrectly |
| 11240 | Fixed | Chart | In some scenarios reloading scripts duplicated global drawing objects |
| 11202 | Fixed | Chart | In some scenarios indicators would not plot with horizontal line plot style |
| 11201 | Fixed | Chart | Chart background color changed when title bar was held |
| 11181 | Fixed | Chart | Focus to chart was not restored after closing data series or indicator window |
| 11167 | Fixed | Chart | Preset Bars to Load for daily bars was not applying when switching time frames by keyboard |
| 11159 | Fixed | Chart | Merge back adjusted was not properly applying |
| 11147 | Fixed | Chart | Bars would stop updating if laptop lid was closed despite it not being in sleep/hibernate mode |
| 11123 | Fixed | Chart | Improved null checks to prevent exceptions |
| 11118 | Fixed | Chart | Anchor points were not plotting on end of displaced indicator |
| 11074 | Fixed | Chart | Using a large displacement on an indicator resulted in a lag |
| 11066 | Fixed | Chart | In some scenarios an indicator could retrace over itself |
| 11105 | Fixed | Chart, DrawingTool | In some scenarios auto scale of drawing objects on secondary panel didn't properly function |
| 10617 | Changed | Chart, DrawingTool | Increased spacing and dash size of lines |
| 11104 | Fixed | Chart, Strategy | In some scenarios drag and drop of strategy plot did not properly function |
| 11215 | Fixed | Continuum, CQG | Properly canceled orders reported cancel failed in some scenarios |
| 11216 | Fixed | Control Center | Orders/Executions tabs filtering by live account included sim101 orders/executions |
| 10973 | Added | Core | Modal on modal windows have a dark overlay applied to better identify the modal window |
| 11254 | Fixed | Database | Automatic rollover failed if triggered while data was loading on start up |
| 11296 | Fixed | DrawingTool | Risk/Reward drawing tool could result in crash with specific configurations |
| 11255 | Fixed | DrawingTool | Data settings were not saved or applied when pressing Enter key |
| 11214 | Fixed | DrawingTool | In some scenarios adding a line to the Trend Channel resulted in freezing |
| 11163 | Fixed | DrawingTool | Remove all drawing tools was applying to all instruments and time frames |
| 11122 | Fixed | DrawingTool | In some scenarios risk-reward tool calculated ticks incorrectly |
| 11111 | Fixed | DrawingTool | Changing primary period on multi-series chart with region could produce an error |
| 11079 | Fixed | eSignal | Index SP500 was not plotting |
| 11206 | Fixed | FX Board | Orders were sorted inconsistently from other order grids |
| 11140 | Fixed | Google, Yahoo | Connection loss was not being reported |
| 11304 | Fixed | Instruments | Importing a stock with a period converted the symbol mapping to use an underscore rather than just the master symbol |
| 11220 | Fixed | Instruments | Stocks imported with invalid characters were not converted and could not be edited |
| 11219 | Fixed | Instruments | Use Instrument Settings template was not reflecting changes to default template |
| 11195 | Fixed | Instruments | Selecting the search button would not search what was typed in |
| 11168 | Fixed | Instruments | Edited default instrument values would reset on restart |
| 11282 | Fixed | Interactive Brokers | In some scenarios targets showed as external orders |
| 11129 | Fixed | Interactive Brokers | Execution markers were not shown on custom instruments |
| 11291 | Fixed | Log | Auto close displayed incorrect time in Log tab |
| 11148 | Fixed | Market Analyzer | In some scenarios the Realized PnL showed incorrect currency symbol |
| 11072 | Fixed | Market Analyzer | Changing the Instrument column to a bar graph resulted in an error |
| 11308 | Fixed | NinjaScript | Auto close disabled strategies on instruments not in auto close list |
| 11262 | Fixed | NinjaScript | ChartControl was null in State.Historical of hosted indicator |
| 11183 | Fixed | NinjaScript | Minimizing and maximizing a chart with text drawn from a script could produce an error |
| 11182 | Fixed | NinjaScript | Changing types after applying indicator to chart would cause errors when opening indicator dialog after recompile |
| 11180 | Fixed | NinjaScript | Saving a file in Visual studio resulted in file being removed from .csproj |
| 11174 | Fixed | NinjaScript | A crash could occur while printing when an output window is open in another workspace |
| 11115 | Fixed | NinjaScript | Drawing objects configured to draw on price panel would move with plot when drag and dropped |
| 11089 | Fixed | NinjaScript | Region could draw past first bar of chart |
| 11062 | Fixed | NinjaScript | In some scenarios RemoveDrawObjects did not properly function |
| 11189 | Fixed | NinjaScript Editor | Renaming an indicator folder resulted in an error |
| 11134 | Fixed | NinjaScript Editor | Right clicking and pasting removed focus from the editor |
| 11223 | Fixed | NinjaScript, Chart | Negative ZOrder resulted in unnecessary OnRender updates |
| 11135 | Fixed | NinjaScript, Chart | On a multi-data series chart with strategies applied to both data series, moving a plot moved both strategy's plots |
| 11297 | Fixed | NinjaScript, DrawingTool | Andrews Pitchfork was not setting extension anchor correctly in NinjaScript |
| 11233 | Fixed | NinjaScript, DrawingTool | Risk-reward was not applying globally when set to do so in a script |
| 11261 | Fixed | Playback | When using Go To the minute and second could not be adjusted using arrows |
| 11244 | Fixed | Playback | ATM targets would stack in playback if 'wrong-side' limit entry orders were placed |
| 11141 | Fixed | Playback | Multi-data series charts would not load renko bars |
| 11166 | Fixed | Playback, Alerts | Alerts would trigger while Playback was paused |
| 11165 | Fixed | Playback, Alerts | Alerts time stamps were reporting as the current time of the PC clock |
| 11248 | Fixed | Playback, NinjaScript | Position.GetUnrealizedProfitLoss() prevented Playback from running |
| 11303 | Fixed | Playback, SuperDOM | Volume column was not updating each second as designed |
| 11258 | Fixed | Strategy | Brush property was not read-only when strategy was enabled |
| 11236 | Fixed | Strategy | Enabling a terminated strategy could result in multiple instances of strategy |
| 10997 | Fixed | Strategy | Account position sync took multiple enables to properly display |
| 11310 | Fixed | Strategy Analyzer | Switching display mode while backtest ran resulted in no chart displayed |
| 11266 | Fixed | Strategy Analyzer | Using both interger and bool properties resulted in an error |
| 11234 | Fixed | Strategy Analyzer | Category order of properties was not properly sorting |
| 11209 | Fixed | Strategy Analyzer | Scrolling results of optimization on a list of instruments could result in an error |
| 11204 | Fixed | Strategy Analyzer | In some scenarios Logs displayed wrong strategy when multiple strategies were in different namespaces |
| 11157 | Fixed | Strategy Analyzer | Sending optimization results to a new Strategy Analyzer was not keeping settings |
| 11132 | Fixed | Strategy Analyzer | Chart Plot lines disappeared when moving multi-series Strategy Analyzer |
| 11131 | Fixed | Strategy Analyzer | Setting optimization graph to performance did not properly list the axis as performance |
| 11120 | Fixed | Strategy Analyzer | In some scenarios indicators region removed when adding other indicators |
| 11272 | Fixed | Strategy Builder | String input did not escape characters |
| 11269 | Changed | Strategy Builder | Unrealized PNL condition was not functioning as expected |
| 11267 | Fixed | Strategy Builder | String inputs with a blank default value were not added into set defaults resulting in an error |
| 11232 | Fixed | Strategy Builder | Setting IsRising regression channel to true resulted in an error |
| 11217 | Fixed | Strategy Builder | In some scenarios Conditions and Actions tabs regenerated after being closed |
| 11169 | Fixed | Strategy Builder | Printing an indicator with bars ago set up would return an unexpected value |
| 11103 | Fixed | Strategy Builder | In some scenarios condition templates with additional data could result in errors |
| 11087 | Fixed | Strategy Builder | In some scenarios text could be selected unexpectedly |
| 11156 | Fixed | Strategy, Chart | Moving a strategy's indicator above panel 1 from another panel would also move the strategy plot to the data series panel |
| 11096 | Fixed | Strategy, Chart | Removing secondary series with strategy applied could leave a blank panel |
| 11192 | Fixed | TD AMERITRADE, NinjaScript | Clicking Close was unable to close out a stop loss order |
| 11150 | Fixed | Tick Replay | Adding an indicator while Tick Replay was loading could result in an error |
| 11125 | Fixed | Tick Replay | Applying an incompatible script wasn't showing an exception in some scenarios |
| 11281 | Fixed | Trade Performance | Journal was not displaying added notes when start and end dates were set to current date |
| 11241 | Fixed | Trade Performance, Workspaces | Display setting was not saving in workspace |
| 11277 | Fixed | Trading Hours | Removing all but one EOD option on the Default 24/7 template resulted in a crash |
| 11249 | Fixed | UI | In some scenarios context menus would get removed before the mouse could reach it |
| 11212 | Fixed | UI | Applying strategy in Control Center was not applying data series presets |
| 11245 | Fixed | Window Linking, NinjaScript | In some scenarios switching instruments on linked tabs resulted in flat indicator plots |
| 11130 | Fixed | Workspaces | Switching workspaces was not placing windows on top of other windows |

**8.0.5.1 Release**

March 8, 2017

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 11346 | Fixed | Strategy | Editing the properties of any existing NinjaScript Strategy on the Control Center would incorrectly reset any DataSeries value back to defaults. This was a result of a fix for issue: 11212 |

**8.0.5.2 Release**

March 15, 2017

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Status** | **Category** | **Comments** |
| 11384 | Fixed | General | Fixed a bug where fresh new installations of NinjaTrader did not get the latest version of the default instrument lists from the server as expected. |
| **Navigation:**  »No topics above this level«  **8.0.0.13 (RC1) Release Notes** | | | | [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout.htm) |

**Release Date**

August 31st, 2016

This release marks our first NinjaTrader 8 Release Candidate.  Since the very first NinjaTrader 8 beta version, we have resolved over 4500 bugs thanks to the ongoing efforts of our beta community, and we feel confident we are closer than ever to a production NinjaTrader 8 release.  If you have installed a NinjaTrader 8 beta version and reported feedback to our support team - thank you!

As this is a release candidate, NinjaTrader 8.0.0.13 is still considered a beta product and we will continue to focus on product quality.  Please continue to report any issues you may encounter to our support staff.   We will monitor the status of this release to determine when we will announce and launch the production release of NinjaTrader 8.

|  |
| --- |
| **Attention MB Trading Users:**Due to limited use and low user feedback during the beta period, we have removed the **MB Trading** adapter from NinjaTrader 8.You may continue to use NinjaTrader 7. |

**Code Breaking Changes**

**Compile Errors**

•The [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object **.Dispose()** method was removed due to technical redundancy.  To remove memory resources from any stroke objects, simply set the stroke to null.

•Removed property **Bars.IsTimebased** -> please use [Bars.BarsType.IsTimeBased](https://ninjatrader.com/es/support/helpGuides/nt8/barstype_istimebased.htm) instead

•**Account.Accounts** was renamed to [Account.All](https://ninjatrader.com/es/support/helpGuides/nt8/all.htm)

**Implementation changes**

•The common signature "isInclude60" used in various [SessionIterator](https://ninjatrader.com/es/support/helpGuides/nt8/sessioniterator.htm) methods was renamed to "includesEndTimeStamp" to be more specific

•Category display order values of standard NinjaTrader**Property Grid Categories** were updated to be more consistent application wide.  These changes could impact any customization you were doing using the [CategoryOrderAttribute](https://ninjatrader.com/es/support/helpGuides/nt8/categoryorderattribute.htm), however, the documentation was also updated to reflect the implementation more accurately and will allow you to use this attribute reliably.

•To assist with transitioning historical order objects to real-time order references, please use the new [GetRealtimeOrder()](https://ninjatrader.com/es/support/helpGuides/nt8/getrealtimeorder.htm) method.

**Notes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Status** | **Issue #** | **Category** | **Comments** |
| Fixed | 10267 | Account Data | Sorting by Commission in Executions tab caused unhandled exception |
| Fixed | 10126 | Account Data, Database | Database caused accounts to show Realized PnL when not connected |
| Fixed | 10123 | Account Data, Rithmic, | Realized PnL only reflected 1 side of the commission |
| Fixed | 10003 | Alerts | Alert Condition Localization issue |
| Fixed | 10164 | Alerts | Exception on opening alerts log window when existing alerts log entry used custom brush |
| Fixed | 10192 | Alerts, DrawingTool | Alerts stop triggering after editing drawing tool anchor via drawing objects window |
| Fixed | 10131 | ATM Strategies | ATM field did not change when submitting ATM from another window on startup |
| Fixed | 10253 | ATM Strategies | ATM is terminated and ATM order cancelled upon order modification failure |
| Fixed | 10279 | ATM Strategies | Modifying ATM Order modified ATM template |
| Fixed | 10217 | Attach Order To Indicator, NinjaScript | Attaching order to indicator via CTRL key resulted in Unhandled Exception |
| Fixed | 10137 | Bars | Crashes could occur when restoring workspaces |
| Fixed | 10220 | Bars | Dates were not showing correctly in New Zealand on daily bars |
| Fixed | 9940 | Bars | Errors could occur when loading data from TD Ameritrade. |
| Fixed | 10132 | Bars | RequestBarSeries1 ERROR while removing workspace in the middle of bars request |
| Fixed | 10105 | Bars | Toggling the Break at EOD with DoNotMerge on large data sets caused blank charts or less bars |
| Fixed | 10036 | Bars, NinjaScript | BarsRequest MergePolicy did not match UseGlobalSettings when using the same policy |
| Changed | 10136 | BarsType | Custom BarType time variable was equal to bars.LastBarTime on new data point |
| Fixed | 10160 | BarsType | Custom BarsType OnDataPoint sometimes had unexpected bid ask data |
| Fixed | 10086 | BarsType | Point and Figure Charts were not calling OnBarUpdate() for each tick or price change |
| Fixed | 10095 | Chart | Chart were allowed to remove last series incorrectly |
| Fixed | 10280 | Chart | Crosshair position showed old position when toggled via hotkey until mouse move |
| Fixed | 10240 | Chart | Drag and drop primary series in same panel caused indicator to jump to primary panel |
| Fixed | 10284 | Chart | Extraneous plot selection point displayed on multi-level indicator as input plot |
| Fixed | 10218 | Chart | Global crosshair render issue at chart boundaries |
| Fixed | 10178 | Chart | Global crosshair incorrect x axis flag time when locked via context menu |
| Fixed | 10156 | Chart | Global draw object was not removed with NinjaScript unless NinjaScript removed manually |
| Fixed | 10097 | Chart | Save Chart Image file name only included the primary data series |
| Fixed | 10242 | Chart | When primary series deleted on multi series chart, instrument selector not updated to new primary |
| Fixed | 10236 | Chart | Z-Order: reloading the historical data reset the z-order |
| Fixed | 10245 | Chart Trader | Errors using ChartTrader with Playback connection |
| Fixed | 10248 | Chart Trader, Templates | Chart Trader settings were ignored when chart template is loaded on an open chart |
| Fixed | 10237 | Chart, Drawing | Unhandled exceptions when moving series panels with GlobalDrawObjects |
| Fixed | 9806 | Chart, Drawing, NinjaScript | Chart could freeze using NinjaScript Draw method/Removing Drawing Tools |
| Fixed | 10117 | Chart, Indicator | System indicators did not load properly after connecting to live data |
| Fixed | 10183 | Chart, Templates | Template caused "An item of the same key has been added" when using two of the same indicator |
| Fixed | 10254 | Chart, Workspaces | InvalidOperationException on restoring chart with template/preset |
| Fixed | 10002 | Commissions | Forex Commissions were factoring Per-Unit instead of Per Lot Size |
| Fixed | 10046 | Control Center | Connection status indicator sometimes did not update |
| Fixed | 10144 | Control Center | Edit strategy dialog was throwing incorrect error |
| Fixed | 10091 | Control Center | Deleted account connection could be disconnected |
| Fixed | 10023 | Control Center, Playback, Strategy | Playback trades performance realized PnL did not match strategies tab of control center |
| Changed | 10111 | Control Center, Strategy | SystemPerformance object was not updating for control center enabled strategies |
| Fixed | 10158 | Control Center, Strategy | Strategy CurrentBars index repored incorrectly after changing parameter and enabling strategy |
| Fixed | 10275 | Control Center, Workspaces | Control Center was incorrectly restored to primary screen when saved maximized |
| Fixed | 10139 | CQG, Workspaces | Workspace charts did not load after connecting to account with no data |
| Fixed | 9909 | Drawing, Strategy Analyzer | Strategy analyzer was not releasing memory when adding indicator that draws objects |
| Fixed | 10167 | DrawingTool | "Remove all drawing objects" did not remove objects unless the originating tab is selected |
| Fixed | 10184 | DrawingTool | Draw object incorrect resize or incorrect anchor on attempting to move past start bar of chart |
| Fixed | 10113 | DrawingTool | Draw objects in future would move when days to load changed |
| Fixed | 10265 | DrawingTool | Draw.Text autoscale was not working correctly |
| Fixed | 10264 | DrawingTool | Draw.Text no alignment behavior did not match NT7 |
| Fixed | 10099 | DrawingTool | Drawing tool template with attach to all charts was drawing on other charts even if cancelled |
| Fixed | 10179 | DrawingTool | Moving global draw object on multi series chart changed anchors incorrectly |
| Fixed | 10061 | DrawingTool | Ray selection points were not lined up on logarithmic y-axis scale |
| Fixed | 10127 | DrawingTool, Playback | Global draw anchors were not consistent in multi-series playback |
| Fixed | 10224 | DrawingTool, Templates | Unable to apply a template more than once to a drawing object |
| Fixed | 10124 | eSignal | ESignal historical tick data timestamps did not match NT7 |
| Fixed | 10214 | FX Board, Hotlist Analyzer, Market Analyzer | Create instrument list context menu item did not work if no instruments loaded |
| Changed | 10208 | Indicator | Consistency updates to indicator error handling |
| Fixed | 10157 | Indicator | BuySellPressure when called from another indicator could cause errors |
| Fixed | 10274 | Indicator | Indicator label did not saved with preset |
| Fixed | 9748 | Instruments | @GER30  CFD Data does not show decimal values |
| Changed | 10145 | Instruments, Yahoo | Yahoo connection is not updating Splits and Dividends. |
| Changed | 10170 | Interactive Brokers | Removed IB Linked Account Support |
| Fixed | 10212 | Interactive Brokers | IB Paper Trading account would not connect on version 954 TWS |
| Fixed | 9993 | Interactive Brokers | Incorrect instrument update price on some instruments |
| Fixed | 9923 | Kinetick | Connection loss loop could occur in some situations |
| Fixed | 10088 | Licensing | FreeTrial Vendor License caused excess lines in Config.xml |
| Fixed | 10209 | Licensing | From and to dates in grid in Vendor Licensing window region formatting |
| Fixed | 10060 | Licensing | Vendor License Addon did not update license messages when changing between vendors |
| Fixed | 10108 | Market Analyzer | Market Analyzer Indicator settings were not recognized when applying template |
| Fixed | 10201 | Market Analyzer | Unable to remove expired instruments from market analyzer in some scenarios |
| Changed | 10196 | Market Analyzer, Workspaces | Custom Market Analyzer Column could not serialize CurrentText |
| Changed | 10180 | MBTrading | Removed MBT Adapter in NinjaTrader 8 |
| Changed | 10106 | NinjaScript | Incorrect sound played when SetProfitTarget target is reached |
| Changed | 10260 | NinjaScript | Renamed a few properties to meet coding guidelines |
| Fixed | 10071 | NinjaScript | CancelOrder() would not cancel historical working orders in State.Realtime |
| Fixed | 10222 | NinjaScript | Errors could occur after deleting indicator and reloading chart |
| Fixed | 9927 | NinjaScript | Draw.Region displacement was from the left of the chart and not from bar 0 |
| Fixed | 10206 | NinjaScript | Expandable properties were not reseting to defaults |
| Fixed | 10177 | NinjaScript | Indicator Error on calling 'SetState' method with tick replay |
| Fixed | 10112 | NinjaScript | IsSuspendedWhileInactive did not work if suspended prior to data feed connection |
| Fixed | 10221 | NinjaScript | Null Stroke object in AddPlot() cuased platform crash |
| Fixed | 10153 | NinjaScript | Unhandled exception if using PasswordBox in Addon |
| Fixed | 10172 | NinjaScript | Update() on multiseries indicator caused primary series OBU called twice |
| Changed | 8410 | NinjaScript Editor | Added Additional Default Snippets |
| Fixed | 10085 | NinjaScript, Orders | Only first identical State.Historical order moved to State.Realtime |
| Fixed | 10233 | NinjaScript, Strategy | Enabling multiple AdoptAccountPosition strategies would result in incorrect popup message |
| Fixed | 10140 | NinjaScript, UI | Strategy Catagories/properties could get out of sequence in strategy and in strategy analyzer |
| Changed | 10162 | Options, Strategy | ConnectionLossHandling was overwritten between State.SetDefaults and State.Configure |
| Fixed | 10148 | Orders, SuperDOM | Cancel all order icon was not visible for simulated orders on SuperDOM |
| Fixed | 10205 | Playback | Simulation accounts could be added unexpecitly while connected to playback |
| Fixed | 10138 | Playback | Exception was thrown unsubscribing data disconnecting playback |
| Fixed | 10202 | Playback | Errors existed around persisting errors to database |
| Fixed | 10142 | Playback | Playback controller date, time and speed sometimes was not visible |
| Fixed | 10211 | Playback | Playback controller end date did not update until platform restart |
| Fixed | 10149 | Playback | Playback manual trade MAE/MFE/ETD values were incorrect |
| Fixed | 9974 | Playback | Sustained playback could resultng in chart rendering error |
| Fixed | 10258 | Playback | Errors could occur connecting to playback on UTC time zone |
| Fixed | 10200 | Skins, UI | Chart Properties Tab Name drop down was barely visible |
| Added | 10087 | Strategy | Strategy concept needed to help users manage historical to live transitioned orders |
| Fixed | 10107 | Strategy | Limit order did not fill if CancelOrder() used on protective order |
| Fixed | 10064 | Strategy | Realized PnL was not updating for strategy with secondary series on strategies tab |
| Fixed | 10128 | Strategy | Strategy template options are not available when editing from strategy tab |
| Fixed | 10194 | Strategy Analyzer | Backtest sometimes ran twice if display was not set to summary or settings |
| Fixed | 10143 | Strategy Analyzer | Platform crash during optimization when accessing null object after State.Terminated |
| Fixed | 10010 | Strategy Analyzer | Platform could crash when using duplicate to new window |
| Fixed | 10181 | Strategy Analyzer | Trades Display "Strategy Column" was blank after duplicated |
| Fixed | 10210 | Strategy Analyzer | Optimization results did not match displays |
| Fixed | 10204 | Strategy Analyzer | Tab context menu items could become disabled incorrectly |
| Fixed | 10263 | Strategy Analyzer | Strategy template was not saving changes after running a backtest |
| Changed | 10118 | Strategy Builder | Allows Strategy Builder Strategies to be manged from NinjaScript Editor |
| Changed | 10101 | Strategy Builder | Could not export Strategy Builder strategy |
| Changed | 10168 | Strategy Builder | Now freezing custom brushes in Strategy Builder to match best practices |
| Changed | 10100 | Strategy Builder | No historical trades taken due to indicators with barsAgo check |
| Fixed | 10103 | Strategy Builder | Strategy Builder could lose reference to candlestick pattern |
| Fixed | 10090 | Strategy Builder | Plot On Chart was calling AddChartIndicator() in wrong state |
| Fixed | 10225 | Strategy Builder | Drawing action category was using wrong name |
| Fixed | 10272 | Strategy Builder | Exception selecting indicator with NinjaScriptProperty value not set |
| Fixed | 10163 | Strategy Builder | Alert message defaulted to same color for foreground/background |
| Fixed | 10262 | Strategy Builder | Did not set indicator Brush properties |
| Fixed | 10271 | Strategy Builder | Strategies with errors could not be deleted |
| Fixed | 10155 | Strategy Builder | Time series could not be compared with Time input or variable in Strategy Builder conditions |
| Fixed | 10082 | Strategy, Trade Performance | Strategy performance calcs had been off |
| Fixed | 10083 | SuperDOM | Order action name localization issue |
| Fixed | 10246 | SuperDOM | SuperDOM > Columns "i" did not give information on the Columns |
| Fixed | 10187 | SuperDOM, Workspaces | SuperDOM Indicator Days to load property reverts to 2 on restore workspace |
| Fixed | 10195 | TD AMERITRADE | Historical daily bars could hold more decimal places than TickSize |
| Fixed | 10285 | Time and Sales | Columns unexpectedly resize when removing/adding columns |
| Fixed | 10203 | UI | Account Tab -> Typing or selecting in account selector did not send to other linked windows |
| Fixed | 10223 | UI | Control Center could sometimes be out of focus after using menus |
| Fixed | 10238 | UI | When selecting the same instrument in the data series window (in the most recent list), the instrument did not add. |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) >  **OnStateChange()** | | | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/marketdeptheventargs.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/setstate.htm) |

**Definition**

An event driven method which is called whenever the script enters a new [State](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm). The **OnStateChange()** method can be used to configure script properties, create one-time behavior when going from historical to real-time, as well as manage clean up resources on termination.

|  |
| --- |
| **Notes**:  •Viewing any UI element which lists NinjaScript classes (such as the Indicators or Strategies window, a chart's Chart Style dropdown menu, etc.) will initialize all classes of that Type when it is opened, which causes each script to enter **State.SetDefaults**, even if it is not actively configured or running in any window. It is important to keep this in mind when adding logic within **State.SetDefaults** in **OnStateChange()**, as this logic will be processed each time the script is initialized. For example, opening the Indicators window will trigger **State.SetDefaults** for all indicators in the system, and closing the Indicators window will trigger**State.Terminated** for all Indicators. In addition, disconnecting or connecting to a data provider can cause State transitions for any currently active scripts. Further discussion of this aspect of the state change model can be found via [*Understanding the lifecycle of your NinjaScript objects*](https://ninjatrader.com/es/support/helpGuides/nt8/understanding_the_lifecycle_of.htm).  •When an indicator is configured on a chart while a Compile is taking place in the NinjaScript Editor, it can appear that the script passes through **State.Terminated**. However, this is the result of a copy of the script being initialized at compile-time, NOT the result of the indicator on the chart being disabled and re-initialized. |

**Related Methods and Properties**

|  |  |
| --- | --- |
| [SetState()](https://ninjatrader.com/es/support/helpGuides/nt8/setstate.htm) | Method is used for changing the State of any running NinjaScript object. |
| [State](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm) | Represents the current progression of the object as it advances from setup, processing data, to termination. |

**Method Return Value**

This method does not return a value.

**Syntax**  
See example below. The NinjaScript wizards automatically generate the method syntax for you.

Possible states are:

|  |  |  |
| --- | --- | --- |
| **State Name** | **This state is called when** | **This state is where you should** |
| State.SetDefaults | **SetDefaults** is always called when displaying objects in a UI list such as the Indicators dialogue window since temporary objects are created for the purpose of UI display | •Keep as lean as possible  •Set default values (pushed to UI) |
| State.Configure | **Configure** is called after a user adds an object to the applied list of objects and presses the OK or Apply button.  This state is called only once for the life of the object. | •Add additional data series via [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm)  •Declare custom resources |
| State.Active | **Active** is called once after the object is configured and is ready to process data (DrawingTools could see multiple calls as internally an object for hit testing is cloned) | •Used for objects such as [Share Service](https://ninjatrader.com/es/support/helpGuides/nt8/share_service.htm) which do not process price series data  •Indicate the object is ready to being processing information |
| State.DataLoaded | **DataLoaded** is called only once after all data series have been loaded. | •Use for logic that needs to access data related objects like Bars, Instruments, BarsPeriod, TradingHours or instantiating indicators  •Notify that all data series have been loaded  •Initialize any class level variables (including custom [Series<T>](https://ninjatrader.com/es/support/helpGuides/nt8/seriest.htm) objects) |
| State.Historical | **Historical**is called once the object begins to process historical data. This state is called once when running an object in real-time. This object is called multiple times when running a backtest optimization and the property [IsInstantiatedOnEachOptimizationIteration](https://ninjatrader.com/es/support/helpGuides/nt8/isinstantiatedoneachoptimizationiteration.htm) is true (default behavior) | •Notify that the object is processing historical data |
| State.Transition | **Transition** is called once as the object has finished processing historical data but before it starts to process realtime data. | •Notify that the indicator or strategy is is transitioning to realtime data  •Prepare realtime related resources |
| State.Realtime | **Realtime** is called once when the object begins to process realtime data. | •Notify that the indicator or strategy is processing realtime data  •Execute realtime related logic |
| State.Terminated | **Terminated** is called once when the object terminates. | •Notify the object is shutting down  •Use to clean up/dispose of resources |

**Active States vs Data Processing States**

After **State.Configure,**each type of NinjaScript type has its own state management system which can be classified under two categories:

•**Active state:**  State.Active

•**Data Processing states:**State.DataLoaded, State.Historical, State.Transition, State.Realtime

The table below lists each NinjaScript type and it's designed state management system:

|  |  |
| --- | --- |
| **NinjaScript Type** | **State Management System** |
| AddOns\* | Active state |
| BarTypes | Active state |
| ChartStyles | Active state |
| DrawingTools | Active state |
| Indicators | Data Processing states |
| ImportTypes | Active state |
| Market Analyzer Columns | Data Processing states |
| OptimizationFitnesses | Active state |
| Optimizers | Active state |
| PerformanceMetrics | Active state |
| ShareServices | Active state |
| Strategies | Data Processing states |
| SuperDOM Columns | Active state |

|  |
| --- |
| **Tips:**  •Resources created in **State.Configure** and not disposed of immediately will be kept and utilized if the NinjaScript object resides in grids (e.g. Strategy tab on Control Center), even if it is not enabled. Try to create resources in**State.Historical** or **State.DataLoaded** instead, if possible.  •**State.Historical** is called multiple times when running a backtest [optimization](https://ninjatrader.com/es/support/helpGuides/nt8/optimize_a_strategy.htm) on a strategy and the property "[IsInstantiatedOnEachOptimizationIteration](https://ninjatrader.com/es/support/helpGuides/nt8/isinstantiatedoneachoptimizationiteration.htm)" is **true** (default behavior).  • Scripts that require [Calculate](https://ninjatrader.com/es/support/helpGuides/nt8/calculate.htm) to be set by the developer must set this property in **State.Historical** in order to ensure that if this script is a child (hosted) that the parent.Calculate property which is adopted by the child is overridden again.  •When instantiating indicators in a [Multi-Series script](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm) in OnStateChange, the input any hosted indicator is running on should be explicitly stated (since a specific [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) is not guaranteed) |

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {     // Calculate once at the end of every single bar     Calculate = Calculate.OnBarClose;         // Add two plots     AddPlot(Brushes.Blue, "Upper"));     AddPlot(Brushes.Orange, "Lower"));   }     else if (State == State.Configure)   {     // Adds a 5-minute Bars object to the strategy and is automatically assigned     // a Bars object index of 1 since the primary data the strategy is run against     // set by the UI takes the index of 0.             AddDataSeries("AAPL", BarsPeriodType.Minute, 5);       } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) >  **IsInstantiatedOnEachOptimizationIteration** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isfilllimitontouch.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/istradinghoursbreaklinevisible.htm) |

**Definition**

Determines if the strategy should be re-instantiated (re-created) after each optimization run when using the [Strategy Analyzer Optimizer](https://ninjatrader.com/es/support/helpGuides/nt8/optimize_a_strategy.htm).

The **default behavior** is to re-instantiate the strategy for each optimization backtest run. However, the process of re-instantiating a strategy requires more time and computer resources to return results, which could impact the amount of time it takes to run an optimization.  When **false**, the strategy is re-used to save time and computer resources.  Under this design, internal properties are reset to default values after each iteration, but it is possible that user-defined properties and other custom resources may carry their state over from the previous iteration into a new backtest run.  To take advantage of performance optimizations, developers may need to reset class level variables in the strategy otherwise unexpected results can occur.

|  |
| --- |
| **Note**:  If you choose to take advantage of the performance benefits during strategy optimization by setting the **IsInstantiatedOnEachOptimizationIteration** property to **false**, any objects you create in your code **MUST** be reset duringthe appropriate**State**within the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm)method.  Please see the example below on "*Manually resetting class level variables to take advantage of Strategy Analyzer optimizer performance benefits*". |

**Property Value**

This property returns **true** if the strategy is not recycled; otherwise, **false**. Default set to **true**.

|  |
| --- |
| **Warning**:  This property should **ONLY** bet set from the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method during **State.SetDefaults** or **State.Configure** |

**Syntax**

IsInstantiatedOnEachOptimizationIteration

|  |
| --- |
| **Tip**:  The default NinjaTrader indicators and strategies have been optimized to take advantage of performance optimizations as their resources are setup >= **State.Configure**.  Please see the default system indicators and strategies for an idea of how you may improve your strategy and indicator performance, or you may also reference the example code below. |

**Examples**

| ns **Using IsInstantiatedOnEachOptimizationIteration to reset class level variables** |
| --- |
| // A custom trades dictionary is created when strategy is instantiated // since we later set "IsInstantiatedOnEachOptimizationIteration" to true, // we are guaranteed to start with a new object on each optimization run private Dictionary<DateTime, string> myTrades = new Dictionary<DateTime, string>();   protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name       = "My Optimization Test 1";     Description = "Demonstrates using IsInstantiatedOnEachOptimizationIteration to reset a class level variable";     Fast       = 10;     Slow       = 25;       // setting to true so our custom trades dictionary is reset on each optimization run (comes with a performance penalty)     // This is the default behavior.     IsInstantiatedOnEachOptimizationIteration = true;   }     else if (State == State.Terminated)   {     // Print the number of trades at the end of the optimization     if (myTrades != null)     {         // if we set "IsInstantiatedOnEachOptimizationIteration" to false (so not using the default of true), the values here would be unexpected         // since the custom trade dictionary was never explicitly reset at the end of each optimization         Print(myTrades.Count);     }   } }   protected override void OnBarUpdate() {   if (CurrentBar < BarsRequiredToTrade)     return;     if (CrossAbove(SMA(Fast), SMA(Slow), 1))   {     EnterLong();     myTrades.Add(Time[0], "long");     }   else if (CrossBelow(SMA(Fast), SMA(Slow), 1))   {     EnterShort();     myTrades.Add(Time[0], "short");   } }   [Range(1, int.MaxValue), NinjaScriptProperty] [Display(Name = "Fast", GroupName = "NinjaScriptStrategyParameters", Order = 0)] public int Fast { get; set; }   [Range(1, int.MaxValue), NinjaScriptProperty] [Display(Name = "Slow", GroupName = "NinjaScriptStrategyParameters", Order = 1)] public int Slow { get; set; } |

| ns **Manually resetting class level variables to take advantage of Strategy Analyzer optimizer performance benefits** | |
| --- | --- |
| // A custom trades dictionary is declared when strategy is first optimized, // but not instantiated until later in State.DataLoaded, private Dictionary<DateTime, string> myTrades;   // examples of other fields which need to be reset private double myDouble; private bool myBool; private DateTime myDateTime; private Order myOrderObject; private Brush myBrushObject; private SMA mySMAIndicator; private Array myIntArray; private List<object> myList; private Series<double> mySeries;   protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "My Optimization Test 2";     Description = "Demonstrates manually resetting a class level variable without re-instantiating the strategy";     Fast = 10;     Slow = 25;           // in this case, we do not need to re-instantiate the strategy after each optimization     // because we are explicitly resetting the custom trade dictionary in State.DataLoaded     // This design of re-using the strategy instance comes with performance benefits     IsInstantiatedOnEachOptimizationIteration = false;   }     else if (State == State.DataLoaded)   {     // re-create custom trade dictionary on each optimization run     // we are guaranteed to start with a new object on each optimization run     if (myTrades != null)       myTrades.Clear();     else       myTrades = new Dictionary<DateTime, string>();           //Any strategy defaults which are maintained do not need to be reset if they are not mutable as the strategy runs.     //Any strategy state that would be mutable after State.SetDefaults needed to be reset for the next run.     myDouble = double.MinValue;     myBool = false;     myDateTime = DateTime.MinValue;     myOrderObject = null;     myBrushObject = null;     mySMAIndicator = SMA(14);           if (myIntArray != null)         Array.Clear(myIntArray, 0, myIntArray.Length);     else         myIntArray = new int[20];           if (myList != null)         myList.Clear();     else         myList = new List<object>();           mySeries = new Series<double>(this);   } }   protected override void OnBarUpdate() {   if (CurrentBar < BarsRequiredToTrade)     return;     if (CrossAbove(SMA(Fast), SMA(Slow), 1))   {     EnterLong();     myTrades.Add(Time[0], "long");     }   else if (CrossBelow(SMA(Fast), SMA(Slow), 1))   {     EnterShort();     myTrades.Add(Time[0], "short");   } }   [Range(1, int.MaxValue), NinjaScriptProperty] [Display(Name = "Fast", GroupName = "NinjaScriptStrategyParameters", Order = 0)] public int Fast { get; set; }   [Range(1, int.MaxValue), NinjaScriptProperty] [Display(Name = "Slow", GroupName = "NinjaScriptStrategyParameters", Order = 1)] public int Slow { get; set; } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [OnFundamentalData()](https://ninjatrader.com/es/support/helpGuides/nt8/onfundamentaldata.htm) >  **FundamentalDataEventArgs** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onfundamentaldata.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/onfundamentaldata.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/onmarketdata.htm) |

**Definition**

Represents a change in fundamental data and is passed as a parameter in the [OnFundamentalData()](https://ninjatrader.com/es/support/helpGuides/nt8/onfundamentaldata.htm) method.

**Methods and Parameters**

|  |  |
| --- | --- |
| DateTimeValue | A [DateTime](http://msdn2.microsoft.com/en-us/library/system.datetime.aspx) value representing the time |
| DoubleValue | A double value representing fundamental data |
| FundamentalDataType | Possible values:    AverageDailyVolume  Beta  CalendarYearHigh  CalendarYearHighDate  CalendarYearLow  CalendarYearLowDate  CurrentRatio  DividendAmount  DividendPayDate  DividendYield  EarningsPerShare  FiveYearsGrowthPercentage  High52Weeks  High52WeeksDate  HistoricalVolatility  Low52Weeks  Low52WeeksDate  MarketCap  NextYearsEarningsPerShare  PercentHeldByInstitutions  PriceEarningsRatio  RevenuePerShare  SharesOutstanding  ShortInterest  ShortInterestRatio  VWAP |
| IsReset | A bool value representing if an UI reset is needed after a manual disconnect. Note: This is only relevant for columns. Whenever this property is true, the UI needs to be reset. |
| LongValue | A long value representing fundamental data |
| ToString() | A string representation of the FundamentalDataEventArgs object |

**Examples**

| ns |
| --- |
| protected override void OnFundamentalData(FundamentalDataEventArgs fundamentalDataUpdate) {     // Print some data to the Output window     if (fundamentalDataUpdate.FundamentalDataType == FundamentalDataType.AverageDailyVolume)         Print("Average Daily Volume = " + fundamentalDataUpdate.LongValue);     else if (fundamentalDataUpdate.FundamentalDataType == FundamentalDataType.PriceEarningsRatio)         Print("P/E Ratio = " + fundamentalDataUpdate.DoubleValue); } |

**Tips**

1.Not all connectivity providers support all FundamentalDataTypes.

2.EarningsPerShare on eSignal is a trailing twelve months value. On IQFeed it is the last quarter's value.

3.RevenuePerShare is a trailing twelve months value.

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [OnFundamentalData()](https://ninjatrader.com/es/support/helpGuides/nt8/onfundamentaldata.htm) >  **FundamentalDataEventArgs** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onfundamentaldata.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/onfundamentaldata.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/onmarketdata.htm) |

**Definition**

Represents a change in fundamental data and is passed as a parameter in the [OnFundamentalData()](https://ninjatrader.com/es/support/helpGuides/nt8/onfundamentaldata.htm) method.

**Methods and Parameters**

|  |  |
| --- | --- |
| DateTimeValue | A [DateTime](http://msdn2.microsoft.com/en-us/library/system.datetime.aspx) value representing the time |
| DoubleValue | A double value representing fundamental data |
| FundamentalDataType | Possible values:    AverageDailyVolume  Beta  CalendarYearHigh  CalendarYearHighDate  CalendarYearLow  CalendarYearLowDate  CurrentRatio  DividendAmount  DividendPayDate  DividendYield  EarningsPerShare  FiveYearsGrowthPercentage  High52Weeks  High52WeeksDate  HistoricalVolatility  Low52Weeks  Low52WeeksDate  MarketCap  NextYearsEarningsPerShare  PercentHeldByInstitutions  PriceEarningsRatio  RevenuePerShare  SharesOutstanding  ShortInterest  ShortInterestRatio  VWAP |
| IsReset | A bool value representing if an UI reset is needed after a manual disconnect. Note: This is only relevant for columns. Whenever this property is true, the UI needs to be reset. |
| LongValue | A long value representing fundamental data |
| ToString() | A string representation of the FundamentalDataEventArgs object |

**Examples**

| ns |
| --- |
| protected override void OnFundamentalData(FundamentalDataEventArgs fundamentalDataUpdate) {     // Print some data to the Output window     if (fundamentalDataUpdate.FundamentalDataType == FundamentalDataType.AverageDailyVolume)         Print("Average Daily Volume = " + fundamentalDataUpdate.LongValue);     else if (fundamentalDataUpdate.FundamentalDataType == FundamentalDataType.PriceEarningsRatio)         Print("P/E Ratio = " + fundamentalDataUpdate.DoubleValue); } |

**Tips**

1.Not all connectivity providers support all FundamentalDataTypes.

2.EarningsPerShare on eSignal is a trailing twelve months value. On IQFeed it is the last quarter's value.

3.RevenuePerShare is a trailing twelve months value.