|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) >  **Indicator** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onnextdatapoint.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/addline.htm) |

The methods and properties covered in this section are unique to custom indicator development.  Indicator configuration properties globally define various behaviors of indicators. All properties have default values and can be overridden by setting them in the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method of the indicator.

|  |
| --- |
| **Tip**:  See also the "[Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm)" section for more method and properties which are shared by NinjaScript types |

**Methods and Properties**

|  |  |
| --- | --- |
| [AddLine()](https://ninjatrader.com/es/support/helpGuides/nt8/addline.htm) | Adds line objects on a chart. |
| [AddPlot()](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm) | Adds plot objects that define how an indicator or strategy data series render on a chart. |
| [BarsRequiredToPlot](https://ninjatrader.com/es/support/helpGuides/nt8/barsrequiredtoplot.htm) | The number of bars on a chart required before the script plots. |
| [DisplayInDataBox](https://ninjatrader.com/es/support/helpGuides/nt8/displayindatabox.htm) | Determines if plot(s) display in the chart data box. |
| [DrawHorizontalGridLines](https://ninjatrader.com/es/support/helpGuides/nt8/drawhorizontalgridlines.htm) | Plots horizontal grid lines on the indicator panel. |
| [DrawOnPricePanel](https://ninjatrader.com/es/support/helpGuides/nt8/drawonpricepanel.htm) | Determines the chart panel the draw objects renders. |
| [DrawVerticalGridLines](https://ninjatrader.com/es/support/helpGuides/nt8/drawverticalgridlines.htm) | Plots vertical grid lines on the indicator panel. |
| [IndicatorBaseConverter](https://ninjatrader.com/es/support/helpGuides/nt8/indicatorbaseconverter.htm) | A custom TypeConverter class handling the designed behavior of an indicator's property descriptor collection. |
| [IsChartOnly](https://ninjatrader.com/es/support/helpGuides/nt8/ischartonly.htm) | If true, any indicator will be only available for charting usage - indicators with this property enabled would for example not be expected to show if called in a SuperDOM or MarketAnalyzer window. |
| [IsSuspendedWhileInactive](https://ninjatrader.com/es/support/helpGuides/nt8/issuspendedwhileinactive.htm) | Prevents real-time market data events from being raised while the indicator's hosting feature is in a state that would be considered suspended and not in immediate use by a user. |
| [PaintPriceMarkers](https://ninjatrader.com/es/support/helpGuides/nt8/paintpricemarkers.htm) | If true, any indicator plot values display price markers in the y-axis. |
| [ShowTransparentPlotsInDataBox](https://ninjatrader.com/es/support/helpGuides/nt8/showtransparentplotsindatabox.htm) | Determines if plot(s) values which are set to a Transparent brush display in the chart data box. |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **AddLine()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/arelinesconfigurable.htm) |

**Definition**

Adds line objects on a chart.

|  |
| --- |
| **Note:**  Lines are **ONLY** visible from the UI property grid when AddLine() is called from **State.SetDefaults**. If your indicator or strategy dynamically adds lines during **State.Configure**, you will **NOT** have an opportunity to select the line or to set the line configuration via the UI. Alternatively, you may use custom public [Brush](https://ninjatrader.com/es/support/helpGuides/nt8/brushes.htm), [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) or value properties which are accessible in the **State.SetDefaults** and pass those values to AddLine() during**State.Configure**. Calling AddLine() in this manner should be reserved for special cases.  Please see the examples below. |

**Methods and Properties**

|  |  |
| --- | --- |
| [AreLinesConfigurable](https://ninjatrader.com/es/support/helpGuides/nt8/arelinesconfigurable.htm) | Determines if the [line](https://ninjatrader.com/es/support/helpGuides/nt8/addline.htm)(s) used in an indicator are configurable from within the indicator dialog window. |
| [Line Class](https://ninjatrader.com/es/support/helpGuides/nt8/line_class.htm) | Objects derived from the Line class are used to characterize how an oscillator line is visually displayed (plotted) on a chart. |
| [Lines](https://ninjatrader.com/es/support/helpGuides/nt8/lines.htm) | A collection holding all of the Line objects that define the visualization characteristics oscillator lines of the indicator. |

**Syntax**

AddLine(Brush brush, double value, string name)  
AddLine(Stroke stroke, double value, string name)

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

**Parameters**

|  |  |
| --- | --- |
| brush | A Brush object used to construct the line |
| name | A string value representing the name of the line |
| stroke | A Stroke object used to construct the line |
| value | A double value representing the value the line will be drawn at |

**Examples**

| ns | **Defining a single UI configurable static line** |
| --- | --- |
|  | protected override void OnStateChange() {         if (State == State.SetDefaults)   {     Name = "Examples Indicator";       // Adds an oscillator line at a value of 30     AddLine(Brushes.Gray, 30, "Lower");   } } |

| ns | **Indicator which dynamically adds a line in State.Configure** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name                 = "Examples Indicator";       // logical property which user can set     UseSpecialMode   = false;     // Default brush selection pushed to the UI     MyBrush = Brushes.Red;   }   else if (State == State.Configure)   {     // if user enables logical property     if (UseSpecialMode)     {         // add line using default selected brush and special line name         AddLine(MyBrush, 40, "My Special Line");     }     else     {         // otherwise use default selected brush and regular line name         AddLine(MyBrush, 60, "My Regular Line");     }   } }     [XmlIgnore] public Brush MyBrush { get; set; }   public bool UseSpecialMode { 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) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **AddPlot()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/lines.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/areplotsconfigurable.htm) |

**Definition**

Adds plot objects that define how an indicator or strategy data series render on a chart. When this method is called to add a plot, an associated [Series<double>](https://ninjatrader.com/es/support/helpGuides/nt8/seriest.htm) object is created held in the [Values](https://ninjatrader.com/es/support/helpGuides/nt8/value.htm) collection.

|  |
| --- |
| **Note:**  Plots are **ONLY** visible from the UI property grid when AddPlot() is called from **State.SetDefaults**. If your indicator or strategy dynamically adds plots during **State.Configure**, you will **NOT** have an opportunity to select the plot or to set the plot configuration via the UI.  Alternatively, you may use custom public [Brush](https://ninjatrader.com/es/support/helpGuides/nt8/brushes.htm), [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm), or **PlotStyle** properties which are accessible in **State.SetDefaults** and pass those values to AddPlot() during**State.Configure**.  Calling AddPlot() in this manner should be reserved for special cases.  Please see the examples below. |

**Methods and Properties**

|  |  |
| --- | --- |
| [ArePlotsConfigurable](https://ninjatrader.com/es/support/helpGuides/nt8/areplotsconfigurable.htm) | Determines if the plot(s) used in an indicator are configurable within the indicator dialog window. |
| [Displacement](https://ninjatrader.com/es/support/helpGuides/nt8/displacement.htm) | An offset value that shifts the visually displayed value of an indicator. |
| [PlotBrushes](https://ninjatrader.com/es/support/helpGuides/nt8/plotbrushes.htm) | Holds an array of color series objects holding historical bar colors. |
| [Plots](https://ninjatrader.com/es/support/helpGuides/nt8/plots.htm) | A collection holding all of the Plot objects that define their visualization characteristics. |

**Syntax**

AddPlot(Brush brush, string name)  
AddPlot(Stroke stroke, PlotStyle plotStyle, string name)

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

**Parameters**

|  |  |
| --- | --- |
| brush | A Brush object used to construct the plot |
| name | A string representing the name of the plot |
| plotStyle | A PlotStyle object used to construct the style of the plot    Possible values:   PlotStyle.Bar PlotStyle.Block PlotStyle.Cross PlotStyle.Dot PlotStyle.Hash PlotStyle.HLine PlotStyle.Line  PlotStyle.PriceBox PlotStyle.Square PlotStyle.TriangleDown PlotStyle.TriangleLeft PlotStyle.TriangleRight PlotStyle.TriangleUp |
| stroke | A Stroke object used to construct the plot |

|  |
| --- |
| **Tips:**  1.We suggest using the NinjaScript wizard to generate your plots.  2.[Plot](https://ninjatrader.com/es/support/helpGuides/nt8/plots.htm) objects **DO NOT** hold the actual script values. They simply define how the script's values are plotted on a chart.  3.A script may calculate multiple values and therefore hold multiple plots to determine the display of each calculated value. Script values are held in the script's [Values](https://ninjatrader.com/es/support/helpGuides/nt8/value.htm) collection.  4.If you script calls AddPlot() multiple times, then multiple Values series are added per the "three value series" example below  5.For [MultiSeries scripts](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm), plots are synched to the primary series of the NinjaScript object. |

**Examples**

| ns | **Indicator using various AddPlot() signatures** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "Examples Indicator";       // Adds a blue line style plot     AddPlot(Brushes.Blue, "myPlot");       // Adds a blue historgram style plot     AddPlot(new Stroke(Brushes.Blue), PlotStyle.Bar, "myPlot");   } } |

| ns | **Indicator which adds three value series** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "Examples Indicator";       // Add three plots and associated Series<double> objects     AddPlot(Brushes.Blue, "PlotA");     // Defines the plot for Values[0]     AddPlot(Brushes.Red, "PlotB");     // Defines the plot for Values[1]     AddPlot(Brushes.Green, "PlotC");   // Defines the plot for Values[2]   } } protected override void OnBarUpdate() {   Values[0][0] = Median[0];   // Blue "Plot A"   Values[1][0] = Low[0];       // Red "Plot B"   Values[2][0] = High[0];     // Green "Plot C" } |

| ns | **Indicator which dynamically adds a plot in State.Configure** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name                 = "Examples Indicator";       // logical property which user can set     UseSpecialMode   = false;     // Default brush selection pushed to the UI     MyBrush = Brushes.Red;   }   else if (State == State.Configure)   {     // if user enables logical property     if (UseSpecialMode)     {         // add plot using default selected brush and special plot name         AddPlot(MyBrush, "My Special Plot");     }     else     {         // otherwise use default selected brush and regular plot name         AddPlot(MyBrush, "My Regular Plot");     }   } }   protected override void OnBarUpdate() {   if (UseSpecialMode)     Value[0] = Close[0] + High[0] / 2;     else Value[0] = Close[0] \* TickSize / 2; }   [XmlIgnore] public Brush MyBrush { get; set; }   public bool UseSpecialMode { 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) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **BarsRequiredToPlot** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/plots.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/displayindatabox.htm) |

**Definition**

The number of bars on a chart required before the script plots.

|  |
| --- |
| **Note**:  This property is **NOT** the same as a minimum number of bars required to calculate the script values.  OnBarUpdate will always start calculating for the first bar on the chart (CurrentBar 0) |

**Property Value**

An int value that represents the minimum number of bars required.

**Syntax**

BarsRequiredToPlot

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {      if (State == State.SetDefaults)      {           BarsRequiredToPlot = 10; // Do not plot until the 11th bar on the chart          AddPlot(Brushes.Orange, "SMA");      }     } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **DisplayInDataBox** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/barsrequiredtoplot.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawhorizontalgridlines.htm) |

**Definition**

Determines if plot(s) display in the chart data box.

**Property Value**

This property returns **true** if the indicator plot(s) values display in the chart data box; 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**

DisplayInDataBox

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         DisplayInDataBox = false;           AddPlot(Brushes.Orange, "SMA");     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **DrawHorizontalGridLines** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/displayindatabox.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawonpricepanel.htm) |

**Definition**

Plots horizontal grid lines on the indicator panel.

|  |
| --- |
| **Note**:  The indicator panel's parent chart has a similar option 'Grid line - horizontal  which if Visible property set to **false**, will override the indicator's local setting if **true**. |

**Property Value**

This property returns **true** if horizontal grid lines are plotted on the indicator panel; otherwise, **false**. Default set to **true**.

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

**Syntax**

DrawHorizontalGridLines

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         DrawHorizontalGridLines = false; // Horizontal grid lines will not plot on the indicator panel             AddPlot(Brushes.Orange, "SMA");     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **DrawOnPricePanel** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawhorizontalgridlines.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawverticalgridlines.htm) |

**Definition**

Determines the chart panel the draw objects renders

**Property Value**

This property returns **true** if the indicator paints draw objects on the price panel; otherwise when false, draw objects are painted on the actual indicator panel itself. Default set to **true**.

|  |
| --- |
| **Warning**:  This property should **ONLY** be set from the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method during **State.SetDefaults.**Dynamically using DrawOnPricePanel in an indicator outside of State.SetDefaults may show issues when working with that indicator through a hosting strategy via [AddChartIndicator()](https://ninjatrader.com/es/support/helpGuides/nt8/addchartindicator.htm). |

**Syntax**

DrawOnPricePanel

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)      {           DrawOnPricePanel = false; // Draw objects now paint on the indicator panel itself           AddPlot(Brushes.Orange, "SMA");      } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **IndicatorBaseConverter Class** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawverticalgridlines.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/ischartonly.htm) |

**Definition**

A custom [TypeConverter](https://msdn.microsoft.com/en-us/library/system.componentmodel.typeconverter%28v=vs.110%29.aspx" \t "_blank) class handling the designed behavior of an indicator's property descriptor collection.  Use this as a base class for any custom **TypeConverter** you are applying to an indicator class.

|  |
| --- |
| **Notes:**  •A working NinjaScript demo can be found through the reference sample on "[Using a TypeConverter to Customize Property Grid Behavior](http://ninjatrader.com/support/forum/showthread.php?t=97919" \t "_blank)"  •When applying the custom converter, you must fully qualify the name (e.g., "NinjaTrader.NinjaScript.Indicators.MyCustomConveter")  •Additional **TypeConverter** information can be found from the [MSDN documentation](https://msdn.microsoft.com/en-us/library/system.componentmodel.typeconverter%28v=vs.110%29.aspx)  •See also [TypeConverterAttribute](https://ninjatrader.com/es/support/helpGuides/nt8/typeconverterattribute.htm)  •For Strategies, see the [StrategyBaseConverter](https://ninjatrader.com/es/support/helpGuides/nt8/strategybaseconverter.htm) class |

**Relevant base methods**

|  |  |
| --- | --- |
| [TypeConverter.GetProperties()](https://msdn.microsoft.com/en-us/library/system.componentmodel.typeconverter.getproperties(v=vs.110).aspx) | When overriding **GetProperties()**, calling base.GetProperties() ensures that all default property grid behavior works as designed |
| [TypeConverter.GetPropertiesSupported()](https://msdn.microsoft.com/en-us/library/system.componentmodel.typeconverter.getpropertiessupported(v=vs.110).aspx) | In your custom converter class, you must override **GetPropertiesSupported()**and return a value of **true** in order for your custom type converter to work |

**Syntax**

public class IndicatorBaseConverter : TypeConverter

|  |
| --- |
| **Warning**:  Failure to apply a type of **IndicatorBaseConverter** on an indicator class can result in unpredictable behavior of the standard NinjaTrader WPF property grid. |

|  |
| --- |
| **Tip**: Common indicator functions like Print() are not available to a type converter instance.  To debug a type converter class, you can use the AddOn [Debug Concepts](https://ninjatrader.com/es/support/helpGuides/nt8/alert_and_debug_concepts.htm) or [attach to a debugger](https://ninjatrader.com/es/support/helpGuides/nt8/visual_studio_debugging.htm) (recommended) |

**Examples**

| ns | |
| --- | --- |
| //This namespace holds Indicators in this folder and is required. Do not change it. namespace NinjaTrader.NinjaScript.Indicators {   // When applying the type converter, you must fully qualify the name   [TypeConverter("NinjaTrader.NinjaScript.Indicators.MyCustomConveter")]   public class MyCustomIndicator : Indicator   {     protected override void OnStateChange()     {         if (State == State.SetDefaults)         {           Name   = "MyCustomIndicator";         }     }       protected override void OnBarUpdate()     {         //Add your custom indicator logic here.     }   }     public class MyCustomConveter : IndicatorBaseConverter   {     // A general TypeConveter method used for converting types     public override PropertyDescriptorCollection GetProperties(ITypeDescriptorContext context, object component, Attribute[] attrs)     {         // sometimes you may need the indicator instance which actually exists on the grid         MyCustomIndicator indicator = component as MyCustomIndicator;           // base.GetProperties ensures we have all the properties (and associated property grid editors)         // NinjaTrader internal logic handles for a given indicator         PropertyDescriptorCollection propertyDescriptorCollection = base.GetPropertiesSupported(context)                 ? base.GetProperties(context, component, attrs) : TypeDescriptor.GetProperties(component, attrs);           if (indicator == null || propertyDescriptorCollection == null)           return propertyDescriptorCollection;           // example of why you may need the instance that exists on the grid....         if (indicator.EntryHandling == EntryHandling.UniqueEntries)         {           // do something in the event a property contains some value...         }           // Loop all of the properties of the indicator         foreach (PropertyDescriptor property in propertyDescriptorCollection)         {           // do something with a specific property             // cannot call Print() here           // but you can call the static Output window "Process()"           NinjaTrader.Code.Output.Process(property.Name, PrintTo.OutputTab1);         }           // must return the collection after making changes         return propertyDescriptorCollection;     }       // Important:  This must return true otherwise the type converter will not be called     public override bool GetPropertiesSupported(ITypeDescriptorContext context)     { return true; }     }   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **IsChartOnly** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/indicatorbaseconverter.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/issuspendedwhileinactive.htm) |

**Definition**

If true, any indicator will be only available for charting usage - indicators with this property enabled would for example not be expected to show if called in a SuperDOM or MarketAnalyzer window.

**Property Value**

This property returns **true** if the indicator can only be used on a chart; otherwise, **false**. Default set to **false**.

|  |
| --- |
| **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**

IsChartOnly

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         IsChartOnly = true; // Allow the indicator to work in charting environment only           } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **IsChartOnly** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/indicatorbaseconverter.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/issuspendedwhileinactive.htm) |

**Definition**

If true, any indicator will be only available for charting usage - indicators with this property enabled would for example not be expected to show if called in a SuperDOM or MarketAnalyzer window.

**Property Value**

This property returns **true** if the indicator can only be used on a chart; otherwise, **false**. Default set to **false**.

|  |
| --- |
| **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**

IsChartOnly

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         IsChartOnly = true; // Allow the indicator to work in charting environment only           } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **IsSuspendedWhileInactive** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ischartonly.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/paintpricemarkers.htm) |

**Definition**

Prevents OnBarUpdate from being raised while the indicators display is not in use.  Enabling this property in your indicator helps save CPU cycles while the indicator is suspended and not in use by a user.  Once the indicator is in a state that would no longer be considered suspended, the historical OnBarUpdate() events will be triggered allowing the indicator to catch up to current real-time values.

Suspension occurs in the following scenarios:

•Minimized Chart

•Minimized Market Analyzer

•Minimized Hot List Analyzer

•Minimized SuperDOM

•Background tabs of above features are considered "minimized"

•Inactive workspaces in the background

|  |
| --- |
| **Note**:  Since events in OnBarUpdate() will not be processed while the indicator is suspended, internal NinjaScript functions such as [Alert()](https://ninjatrader.com/es/support/helpGuides/nt8/alert.htm), [PlaySound()](https://ninjatrader.com/es/support/helpGuides/nt8/playsound.htm), [Share()](https://ninjatrader.com/es/support/helpGuides/nt8/share.htm), [Print()](https://ninjatrader.com/es/support/helpGuides/nt8/print.htm), etc - or any other method that would be used to notify a user of activity will **NOT** be processed until the indicator is un-suspended. |

**Scenarios where suspension will not occur**

The **IsSuspendedWhileInactive** property will be ignored and real-time events will be processed as normal under the following cases:

•Indicators running in [Automated NinjaScript Strategies](https://ninjatrader.com/es/support/helpGuides/nt8/running_a_ninjascript_strategy.htm)

•Indicators which have [manually configured alerts](https://ninjatrader.com/es/support/helpGuides/nt8/alerts_dialog.htm)

•Indicators which have been [manually attached to orders](https://ninjatrader.com/es/support/helpGuides/nt8/attachingorderstoindicators.htm)

**Property Value**

This property returns **true** if indicator can take advantage of suspension optimization; otherwise, **false**. Default set to **false**.

|  |
| --- |
| **Note**:  This property is overridden to "**true**" automatically by the [NinjaScript Code Wizard](https://ninjatrader.com/es/support/helpGuides/nt8/ns_wizard.htm).  You will need to remove the property to return to the default value or manually set it to false to disable this behavior |

|  |
| --- |
| **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**

IsSuspendedWhileInactive

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         IsSuspendedWhileInactive = true;     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **PaintPriceMarkers** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/issuspendedwhileinactive.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/showtransparentplotsindatabox.htm) |

**Definition**

If true, any indicator plot values display price markers in the y-axis.

**Property Value**

This property returns **true** if the indicator plot values display in the y-axis; 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**

PaintPriceMarkers

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         PaintPriceMarkers = true; // Indicator plots values display in the y-axis             AddPlot(Brushes.Orange, "SMA");     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **ShowTransparentPlotsInDataBox** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/paintpricemarkers.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/market_analyzer_column.htm) |

**Definition**

Determines if plot(s) values which are set to a Transparent brush display in the chart data box.  The default behavior is to hide any plots which have been configured as a Transparent brush, however this behavior can be changed by setting **ShowTransparentPlotsInDataBox** to **true** on the indicator.

**Property Value**

This property returns **true** if transparent indicator plot(s) values display in the chart data box; otherwise, **false**. Default set to **false**.

|  |
| --- |
| **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**

ShowTransparentPlotsInDataBox

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         ShowTransparentPlotsInDataBox = true;           AddPlot(Brushes.Transparent, "MyPlot");     } } | |
| **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) >  **AddChartIndicator()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/strategy_account.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/addperformancemetric.htm) |

**Definition**

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

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

**Method Return Value**

This method does not return a value.

**Syntax**  
AddChartIndicator(IndicatorBase indicator)

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

**Parameters**

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

**Examples**

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

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

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.DataLoaded)   {     // Charts a 20 period simple moving average to the chart     AddChartIndicator(SMA(20));   } }   protected override void OnBarUpdate() {     // call SMA() historically to ensure the indicator processes its historical states as well   double sma = SMA(20)[0]; } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **AddLine()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/arelinesconfigurable.htm) |

**Definition**

Adds line objects on a chart.

|  |
| --- |
| **Note:**  Lines are **ONLY** visible from the UI property grid when AddLine() is called from **State.SetDefaults**. If your indicator or strategy dynamically adds lines during **State.Configure**, you will **NOT** have an opportunity to select the line or to set the line configuration via the UI. Alternatively, you may use custom public [Brush](https://ninjatrader.com/es/support/helpGuides/nt8/brushes.htm), [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) or value properties which are accessible in the **State.SetDefaults** and pass those values to AddLine() during**State.Configure**. Calling AddLine() in this manner should be reserved for special cases.  Please see the examples below. |

**Methods and Properties**

|  |  |
| --- | --- |
| [AreLinesConfigurable](https://ninjatrader.com/es/support/helpGuides/nt8/arelinesconfigurable.htm) | Determines if the [line](https://ninjatrader.com/es/support/helpGuides/nt8/addline.htm)(s) used in an indicator are configurable from within the indicator dialog window. |
| [Line Class](https://ninjatrader.com/es/support/helpGuides/nt8/line_class.htm) | Objects derived from the Line class are used to characterize how an oscillator line is visually displayed (plotted) on a chart. |
| [Lines](https://ninjatrader.com/es/support/helpGuides/nt8/lines.htm) | A collection holding all of the Line objects that define the visualization characteristics oscillator lines of the indicator. |

**Syntax**

AddLine(Brush brush, double value, string name)  
AddLine(Stroke stroke, double value, string name)

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

**Parameters**

|  |  |
| --- | --- |
| brush | A Brush object used to construct the line |
| name | A string value representing the name of the line |
| stroke | A Stroke object used to construct the line |
| value | A double value representing the value the line will be drawn at |

**Examples**

| ns | **Defining a single UI configurable static line** |
| --- | --- |
|  | protected override void OnStateChange() {         if (State == State.SetDefaults)   {     Name = "Examples Indicator";       // Adds an oscillator line at a value of 30     AddLine(Brushes.Gray, 30, "Lower");   } } |

| ns | **Indicator which dynamically adds a line in State.Configure** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name                 = "Examples Indicator";       // logical property which user can set     UseSpecialMode   = false;     // Default brush selection pushed to the UI     MyBrush = Brushes.Red;   }   else if (State == State.Configure)   {     // if user enables logical property     if (UseSpecialMode)     {         // add line using default selected brush and special line name         AddLine(MyBrush, 40, "My Special Line");     }     else     {         // otherwise use default selected brush and regular line name         AddLine(MyBrush, 60, "My Regular Line");     }   } }     [XmlIgnore] public Brush MyBrush { get; set; }   public bool UseSpecialMode { 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) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **AddPlot()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/lines.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/areplotsconfigurable.htm) |

**Definition**

Adds plot objects that define how an indicator or strategy data series render on a chart. When this method is called to add a plot, an associated [Series<double>](https://ninjatrader.com/es/support/helpGuides/nt8/seriest.htm) object is created held in the [Values](https://ninjatrader.com/es/support/helpGuides/nt8/value.htm) collection.

|  |
| --- |
| **Note:**  Plots are **ONLY** visible from the UI property grid when AddPlot() is called from **State.SetDefaults**. If your indicator or strategy dynamically adds plots during **State.Configure**, you will **NOT** have an opportunity to select the plot or to set the plot configuration via the UI.  Alternatively, you may use custom public [Brush](https://ninjatrader.com/es/support/helpGuides/nt8/brushes.htm), [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm), or **PlotStyle** properties which are accessible in **State.SetDefaults** and pass those values to AddPlot() during**State.Configure**.  Calling AddPlot() in this manner should be reserved for special cases.  Please see the examples below. |

**Methods and Properties**

|  |  |
| --- | --- |
| [ArePlotsConfigurable](https://ninjatrader.com/es/support/helpGuides/nt8/areplotsconfigurable.htm) | Determines if the plot(s) used in an indicator are configurable within the indicator dialog window. |
| [Displacement](https://ninjatrader.com/es/support/helpGuides/nt8/displacement.htm) | An offset value that shifts the visually displayed value of an indicator. |
| [PlotBrushes](https://ninjatrader.com/es/support/helpGuides/nt8/plotbrushes.htm) | Holds an array of color series objects holding historical bar colors. |
| [Plots](https://ninjatrader.com/es/support/helpGuides/nt8/plots.htm) | A collection holding all of the Plot objects that define their visualization characteristics. |

**Syntax**

AddPlot(Brush brush, string name)  
AddPlot(Stroke stroke, PlotStyle plotStyle, string name)

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

**Parameters**

|  |  |
| --- | --- |
| brush | A Brush object used to construct the plot |
| name | A string representing the name of the plot |
| plotStyle | A PlotStyle object used to construct the style of the plot    Possible values:   PlotStyle.Bar PlotStyle.Block PlotStyle.Cross PlotStyle.Dot PlotStyle.Hash PlotStyle.HLine PlotStyle.Line  PlotStyle.PriceBox PlotStyle.Square PlotStyle.TriangleDown PlotStyle.TriangleLeft PlotStyle.TriangleRight PlotStyle.TriangleUp |
| stroke | A Stroke object used to construct the plot |

|  |
| --- |
| **Tips:**  1.We suggest using the NinjaScript wizard to generate your plots.  2.[Plot](https://ninjatrader.com/es/support/helpGuides/nt8/plots.htm) objects **DO NOT** hold the actual script values. They simply define how the script's values are plotted on a chart.  3.A script may calculate multiple values and therefore hold multiple plots to determine the display of each calculated value. Script values are held in the script's [Values](https://ninjatrader.com/es/support/helpGuides/nt8/value.htm) collection.  4.If you script calls AddPlot() multiple times, then multiple Values series are added per the "three value series" example below  5.For [MultiSeries scripts](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm), plots are synched to the primary series of the NinjaScript object. |

**Examples**

| ns | **Indicator using various AddPlot() signatures** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "Examples Indicator";       // Adds a blue line style plot     AddPlot(Brushes.Blue, "myPlot");       // Adds a blue historgram style plot     AddPlot(new Stroke(Brushes.Blue), PlotStyle.Bar, "myPlot");   } } |

| ns | **Indicator which adds three value series** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "Examples Indicator";       // Add three plots and associated Series<double> objects     AddPlot(Brushes.Blue, "PlotA");     // Defines the plot for Values[0]     AddPlot(Brushes.Red, "PlotB");     // Defines the plot for Values[1]     AddPlot(Brushes.Green, "PlotC");   // Defines the plot for Values[2]   } } protected override void OnBarUpdate() {   Values[0][0] = Median[0];   // Blue "Plot A"   Values[1][0] = Low[0];       // Red "Plot B"   Values[2][0] = High[0];     // Green "Plot C" } |

| ns | **Indicator which dynamically adds a plot in State.Configure** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name                 = "Examples Indicator";       // logical property which user can set     UseSpecialMode   = false;     // Default brush selection pushed to the UI     MyBrush = Brushes.Red;   }   else if (State == State.Configure)   {     // if user enables logical property     if (UseSpecialMode)     {         // add plot using default selected brush and special plot name         AddPlot(MyBrush, "My Special Plot");     }     else     {         // otherwise use default selected brush and regular plot name         AddPlot(MyBrush, "My Regular Plot");     }   } }   protected override void OnBarUpdate() {   if (UseSpecialMode)     Value[0] = Close[0] + High[0] / 2;     else Value[0] = Close[0] \* TickSize / 2; }   [XmlIgnore] public Brush MyBrush { get; set; }   public bool UseSpecialMode { 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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **AllowRemovalOfDrawObjects** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/brushes.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/backbrush.htm) |

**Definition**

Determines if programmatically drawn [DrawObjects](https://ninjatrader.com/es/support/helpGuides/nt8/drawingtools_drawobjects.htm) are allowed to remove manually from the chart

**Property Value**

When set to **true**, the draw objects from the indicator or strategy can be deleted from the chart manually by a user. If **false**, draw objects from the indicator or strategy can only be removed from the chart if the script removes the drawing object, or the script is terminates.  Default set to **false**.

**Syntax**

AllowRemovalOfDrawObjects

**Examples**

|  |  |
| --- | --- |
| ns |  |
| protected override void OnStateChange() {     Add(new Plot(Brushes.Orange, "SMA"));     AllowRemovalOfDrawObjects = true; // Draw objects can be removed separately from the script } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) > [AddLine()](https://ninjatrader.com/es/support/helpGuides/nt8/addline.htm) >  **AreLinesConfigurable** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/addline.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/addline.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/line_class.htm) |

**Definition**

Determines if the [line](https://ninjatrader.com/es/support/helpGuides/nt8/addline.htm)(s) used in an indicator are configurable from within the indicator dialog window.

**Property Value**

A bool which **true** if any indicator line(s) are configurable; otherwise, **false**. Default set to **true**.

**Syntax**

AreLinesConfigurable

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         AddLine(Brushes.Gray, 30, "Lower");         AreLinesConfigurable = false; // Indicator lines are not configurable     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) > [AddPlot()](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm) >  **ArePlotsConfigurable** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/displacement.htm) |

**Definition**

Determines if the plot(s) used in an indicator are configurable within the indicator dialog window.

**Property Value**

A bool which returns **true** if any indicator plot(s) are configurable; otherwise, **false**. Default set to **true**.

**Syntax**

ArePlotsConfigurable

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         AddPlot(Brushes.Orange, "SMA");         ArePlotsConfigurable = false; // Plots are not configurable in the indicator dialog     } } | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **BackBrush** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/allowremovalofdrawobjects.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/backbrushall.htm) |

**Definition**

Sets the brush used for painting the chart panel's background color for the current bar.

|  |
| --- |
| **Note**: This property will only set the back color for the panel the indicator is running.  To set background color for all panels, please see the [BackBrushAll](https://ninjatrader.com/es/support/helpGuides/nt8/backbrushall.htm) property. |

**Property Value**

A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx" \t "_blank) object that represents the color of the current chart bar.

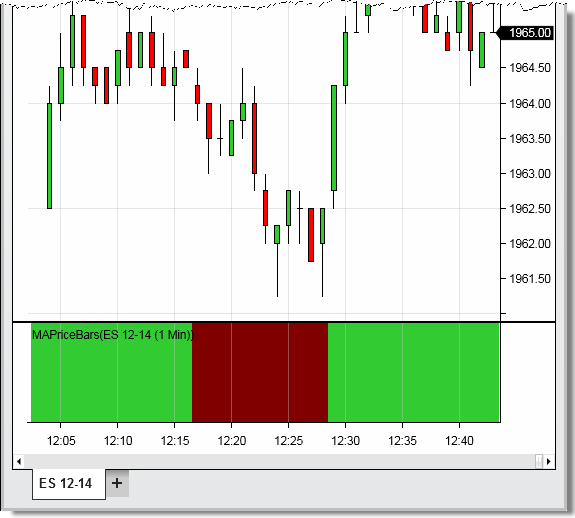
**Syntax**

BackBrush

|  |
| --- |
| **Warning**:  You may have up to 65,535 unique BackBrush instances, therefore, using [static predefined brushes](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_brushes.htm) should be favored.  Alternatively,  in order to use fewer brushes, please try to cache your custom brushes until a new brush would actually need to be created. |

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {     // Sets the chart panel back color to pale green     BackBrush = Brushes.PaleGreen;       // Sets the back color to to null which will use the default color set in the chart properties dialog window     BackBrush = null;       // Sets the back color to maroon when the closing price is less than the 20 period SMA // and to lime green when above (see image below)     BackBrush = SMA(20)[0] >= Close[0] ? Brushes.Maroon : Brushes.LimeGreen; } |



|  |  |
| --- | --- |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **BackBrushAll** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/backbrush.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/backbrushes.htm) |

**Definition**

A collection of prior back brushes used for the background colors for all chart panels.

**Property Value**

A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx" \t "_blank) object that represents the color of the current chart bar.

|  |
| --- |
| **Tip**:  To reset the Chart background color to the default background color property, set the **BackBrush** to null for that bar. |

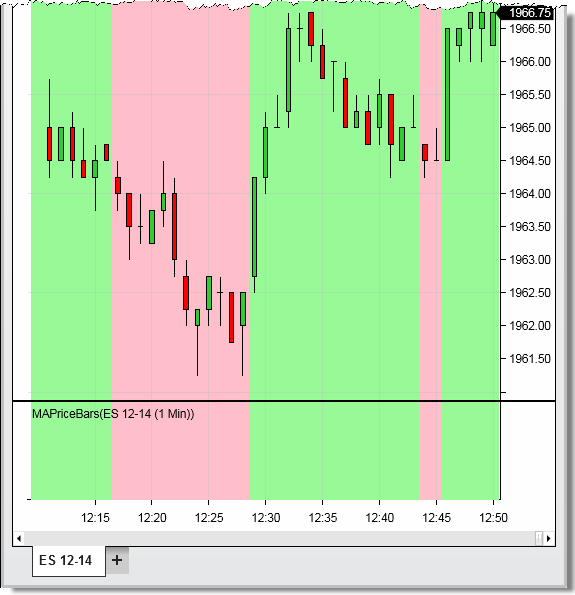
**Syntax**

BackBrushAll

|  |
| --- |
| **Warning**:  You may have up to 65,535 unique BackBrushAll instances, therefore, using [static predefined brushes](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_brushes.htm) should be favored.  Alternatively,  in order to use fewer brushes, please try to cache your custom brushes until a new brush would actually need to be created. |

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {     // Sets the back color to pale green     BackBrushAll = Brushes.PaleGreen;       // Sets the back color to null to use the default color set in the chart properties dialog window     BackBrushAll = null;       // Sets the back color to pink when the closing price is less than the 20 period SMA     // and to lime green when above (see image below)     BackBrushAll = SMA(20)[0] >= Close[0] ? Brushes.Pink : Brushes.PaleGreen; } |



|  |  |
| --- | --- |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **BackBrushes** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/backbrushall.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/backbrushesall.htm) |

**Definition**

A collection of prior back brushes used for the background colors of the chart panel.

**Property Value**

A brush series type object. Accessing this property via an index value [int *barsAgo*] returns a [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx" \t "_blank) object representing the color of the background color on the referenced bar.

**Syntax**

BackBrushes

BackBrushes[int *barsAgo*]

|  |
| --- |
| **Warning**:  You may have up to 65,535 unique BackBrushes instances, therefore, using [static predefined brushes](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_brushes.htm) should be favored.  Alternatively,  in order to use fewer brushes, please try to cache your custom brushes until a new brush would actually need to be created. |

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     if (CurrentBar < 1)         return;       // Sets the color of the background on the current bar as blue     BackBrushes[0] = Brushes.Blue;       // Sets the color of the background on the previous bar as orange     BackBrushes[1] = Brushes.Orange; } | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **BackBrushesAll** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/backbrushes.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/barbrush.htm) |

**Definition**

A collection of historical brushes used for the background colors for all chart panels.

**Property Value**

A brush series type object. Accessing this property via an index value [int *barsAgo*] returns a [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx" \t "_blank) object representing the color of the background color on the referenced bar for all chart panels.

**Syntax**

BackBrushesAll  
BackBrushesAll[int *barsAgo*]

|  |
| --- |
| **Warning**:  You may have up to 65,535 unique BackBrushAll instances, therefore, using [static predefined brushes](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_brushes.htm) should be favored.  Alternatively,  in order to use fewer brushes, please try to cache your custom brushes until a new brush would actually need to be created. |

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     if (CurrentBar < 1)         return;       // Sets the color of the background on the current bar as blue on all chart panels.     BackBrushesAll[0] = Brushes.Blue;       // Sets the color of the background on the previous bar as orange on all chart panels.     BackBrushesAll[1] = Brushes.Orange; } | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **BarBrush** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/backbrushesall.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/barbrushes.htm) |

**Definition**

Sets the brush used for painting the color of a price bar's body.

**Property Value**

A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) object that represents the color of this price bar.

|  |
| --- |
| **Tip**: To set the price bar color to an empty color which uses the default bar color property, set the **BackBrush** to null for that bar. |

**Syntax**

BarBrush

|  |
| --- |
| **Warning**:  You may have up to 65,535 unique BarBrush instances, therefore, using [static predefined brushes](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_brushes.htm) should be favored.  Alternatively,  in order to use fewer brushes, please try to cache your custom brushes until a new brush would actually need to be created. |

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Sets the bar color to yellow     BarBrush = Brushes.Yellow;       // Sets the brush used for the bar color to its default color as defined in the chart properties dialog     BarBrush = null;       // Sets the bar color to yellow if the 20 SMA is above the 50 SMA and the closing     // price is above the 20 SMA (see image below)     if (SMA(20)[0] > SMA(50)[0] && Close[0] > SMA(20)[0])         BarBrush = Brushes.Yellow; } | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **BarBrushes** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/barbrush.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/candleoutlinebrush.htm) |

**Definition**

A collection of historical brushes used for painting the color of a price bar's body.

**Property Value**

A brush series type object. Accessing this property via an index value [int *barsAgo*] returns a [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx" \t "_blank) object representing the referenced bar's color.

|  |
| --- |
| **Note**: This will only return the color of a bar in which an explicit color overwrite was used. Otherwise it will return null. |

**Syntax**

BarBrushes  
BarBrushes[int *barsAgo*]

|  |
| --- |
| **Warning**:  You may have up to 65,535 unique BarBrushes instances, therefore, using [static predefined brushes](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_brushes.htm) should be favored.  Alternatively,  in order to use fewer brushes, please try to cache your custom brushes until a new brush would actually need to be created. |

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     if (CurrentBar < 1)         return;       // Sets the color of the current bar to blue.     BarBrushes[0] = Brushes.Blue;       // Sets the color of the previous bar to orange.     BarBrushes[1] = Brushes.Orange;   } | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **CandleOutlineBrush** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/barbrushes.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/candleoutlinebrushes.htm) |

**Definition**

Sets the outline Brush of a candlestick.

**Property Value**

A [brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) object that represents the color of this price bar.

**Syntax**

CandleOutlineBrush

|  |
| --- |
| **Warning**:  You may have up to 65,535 unique CandleOutlineBrushes instances, therefore, using [static predefined brushes](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_brushes.htm) should be favored.  Alternatively,  in order to use fewer brushes, please try to cache your custom brushes until a new brush would actually need to be created. |

**Examples**

| ns | |
| --- | --- |
| // Sets the candle outline color to black CandleOutlineBrush = Brushes.Black; | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **CandleOutlineBrushes** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/candleoutlinebrush.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawingtools_drawobjects.htm) |

**Definition**

A collection of historical outline brushes for candlesticks.

**Property Value**

A brush series type object. Accessing this property via an index value [int *barsAgo*] returns a [brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx" \t "_blank) structure representing the referenced bar's outline color.

|  |
| --- |
| **Note**: This will only return the color of a candlestick outline in which an explicit color overwrite was used. Otherwise it will return null. |

**Syntax**

CandleOutlineBrushes  
CandleOutlineBrushes[int *barsAgo*]

|  |
| --- |
| **Warning**:  You may have up to 65,535 unique CandleOutlineBrushes instances, therefore, using [static predefined brushes](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_brushes.htm) should be favored.  Alternatively,  in order to use fewer brushes, please try to cache your custom brushes until a new brush would actually need to be created. |

**Examples**

| ns | |
| --- | --- |
| // Sets the outline color of the current bar to black. CandleOutlineBrushes[0] = Brushes.Black;   // Sets the outline color of the previous bar to blue. CandleOutlineBrushes[1] = Brushes.Blue; | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **DisplayInDataBox** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/barsrequiredtoplot.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawhorizontalgridlines.htm) |

**Definition**

Determines if plot(s) display in the chart data box.

**Property Value**

This property returns **true** if the indicator plot(s) values display in the chart data box; 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**

DisplayInDataBox

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         DisplayInDataBox = false;           AddPlot(Brushes.Orange, "SMA");     } } | |
| **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) >  **IsOverlay** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isautoscale.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chart.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isseparatezorder.htm) |

**Definition**

Determines if indicator plot(s) are drawn on the chart panel over top of price.  Setting this value to true will also allow an Indicator to be used as a [SuperDOM Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_indicators_superdom.htm).

**Property Value**

This property returns **true** if any indicator plot(s) are drawn on the chart panel; otherwise, **false**. Default set to **false**.

|  |
| --- |
| **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**

IsOverlay

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         IsOverlay = true; // Indicator plots are drawn on the chart panel on top of price             AddPlot(Brushes.Orange, "SMA");     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) > [AddPlot()](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm) >  **PlotBrushes** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/displacement.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/plots.htm) |

**Definition**

Holds an array of color series objects holding historical bar colors. A color series object is added to this array when calling the [AddPlot()](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm) method in a custom Indicator for plots. Its purpose is to provide access to the color property of all bars.

**Property Value**

An array of color series objects.

**Syntax**  
PlotBrushes[int *PlotIndex*][int *barsAgo*]

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if(State == State.SetDefaults)   {     Name = "Example Indicator";           // Add two plots       AddPlot(Brushes.Blue, "Upper");       AddPlot(Brushes.Orange, "Lower");     } }   protected override void OnBarUpdate() {     // Sets values to our two plots     Upper[0] = SMA(High, 20)[0];     Lower[0] = SMA(Low, 20)[0];       // Color the Upper plot based on plot value conditions     if (IsRising(Upper))         PlotBrushes[0][0] = Brushes.Blue;     else if (IsFalling(Upper))         PlotBrushes[0][0] = Brushes.Red;     else         PlotBrushes[0][0] = Brushes.Yellow;       // Color the Lower plot based on plot value conditions     if (IsRising(Lower))         PlotBrushes[1][0] = Brushes.Blue;     else if (IsFalling(Lower))         PlotBrushes[1][0] = Brushes.Red;     else         PlotBrushes[1][0] = Brushes.Yellow; }   public Series<double> Upper {   get { return Values[0]; } }   public Series<double> Lower {   get { return Values[1]; } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) > [AddPlot()](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm) >  **Plots** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/plotbrushes.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/barsrequiredtoplot.htm) |

**Definition**

A collection holding all of the Plot objects that define their visualization characteristics.

**Property Value**

A collection of Plot objects.

**Syntax**

Plots[int *index*]

|  |
| --- |
| **Note**: The example code below will change the color of an entire plot series. See [PlotBrushes](https://ninjatrader.com/es/support/helpGuides/nt8/plotbrushes.htm) for information on changing only specific segments of a plot instead. |

**Example**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if(State == State.SetDefaults)   {       Name = "Examples Indicator";       // Lines are added to the Lines collection in order       AddPlot(Brushes.Orange, "Plot1"); // Stored in Plots[0]       AddPlot(Brushes.Blue, "Plot2");   // Stored in Plots[1]     } }   // Dynamically change the primary plot's color based on the indicator value protected override void OnBarUpdate() {   if (Value[0] > 70)   {     Plots[0].Brush = Brushes.Blue;     Plots[0].Width = 2;   }   else   {     Plots[0].Brush = Brushes.Red;     Plots[0].Width = 2;   } } | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **BarBrushes** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/barbrush.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/candleoutlinebrush.htm) |

**Definition**

A collection of historical brushes used for painting the color of a price bar's body.

**Property Value**

A brush series type object. Accessing this property via an index value [int *barsAgo*] returns a [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx" \t "_blank) object representing the referenced bar's color.

|  |
| --- |
| **Note**: This will only return the color of a bar in which an explicit color overwrite was used. Otherwise it will return null. |

**Syntax**

BarBrushes  
BarBrushes[int *barsAgo*]

|  |
| --- |
| **Warning**:  You may have up to 65,535 unique BarBrushes instances, therefore, using [static predefined brushes](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_brushes.htm) should be favored.  Alternatively,  in order to use fewer brushes, please try to cache your custom brushes until a new brush would actually need to be created. |

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     if (CurrentBar < 1)         return;       // Sets the color of the current bar to blue.     BarBrushes[0] = Brushes.Blue;       // Sets the color of the previous bar to orange.     BarBrushes[1] = Brushes.Orange;   } | |
| **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) >  **IsAutoScale** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/formatpricemarker.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chart.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isoverlay.htm) |

**Definition**

If true, the drawing tool will call [CalculateMinMax()](https://ninjatrader.com/es/support/helpGuides/nt8/oncalculateminmax.htm) in order to determine the drawing tool's [MinValue](https://ninjatrader.com/es/support/helpGuides/nt8/minvalue.htm) and [MaxValue](https://ninjatrader.com/es/support/helpGuides/nt8/maxvalue.htm) value used to scale the Y-axis of the chart.

**Property Value**

This property returns **true** if the drawing tool plot(s) are included in the y-scale; otherwise, **false**. Default set to **false**.

|  |
| --- |
| **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**

IsAutoScale

**Example**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {             Name                 = "Example Indicator";             // set this to true to call CalculateMinMix() to ensure drawing tool is fully rendered in chart scale     IsAutoScale = true;     }   else if (State == State.Configure)   {   } } | |
| **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) >  **ScaleJustification** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isseparatezorder.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chart.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) |

**Definition**

Determines which scale an indicator will be plotted on.

|  |
| --- |
| **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** |

**Property Value**

This property returns a ScaleJustification value of either:

NinjaTrader.Gui.Charts.ScaleJustification.Left;  
NinjaTrader.Gui.Charts.ScaleJustification.Overlay;  
NinjaTrader.Gui.Charts.ScaleJustification.Right;

**Syntax**

ScaleJustification

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {         if (State == State.SetDefaults)   {     Name = "Examples Indicator";         // force "My Plot" to be plotted on the left scale     ScaleJustification = ScaleJustification.Left;     }   else if (State == State.Configure)   {                 AddPlot(Brushes.Orange, "My Plot");   } } | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.AndrewsPitchfork()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/andrewspitchfork.htm) |

**Definition**

Draws an Andrew's Pitchfork.

**Method Return Value**

An [AndrewsPitchfork](https://ninjatrader.com/es/support/helpGuides/nt8/andrewspitchfork.htm) object that represents the draw object.

**Syntax**

Draw.AndrewsPitchfork(NinjaScriptBase owner, string tag, bool isAutoScale, int anchor1BarsAgo, double anchor1Y, int anchor2BarsAgo, double anchor2Y, int anchor3BarsAgo, double anchor3Y, Brush brush, DashStyleHelper dashStyle, int width)  
Draw.AndrewsPitchfork(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime anchor1Time, double anchor1Y, DateTime anchor2Time, double anchor2Y, DateTime anchor3Time, double anchor3Y, Brush brush, DashStyleHelper dashStyle, int width)  
Draw.AndrewsPitchfork(NinjaScriptBase owner, string tag, bool isAutoScale, int anchor1BarsAgo, double anchor1Y, int anchor2BarsAgo, double anchor2Y, int anchor3BarsAgo, double anchor3Y, bool isGlobal, string templateName)  
Draw.AndrewsPitchfork(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime anchor1Time, double anchor1Y, DateTime anchor2Time, double anchor2Y, DateTime anchor3Time, double anchor3Y, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale |
| anchor1BarsAgo | The number of bars ago (x value) of the 1st anchor point |
| anchor1Time | The time of the 1st anchor point |
| anchor1Y | The y value of the 1st anchor point |
| anchor2BarsAgo | The number of bars ago (x value) of the 2nd anchor point |
| anchor2Time | The time of the 2nd anchor point |
| anchor2Y | The y value of the 2nd anchor point |
| anchor3BarsAgo | The number of bars ago (x value) of the 3rd anchor point |
| anchor3Time | The time of the 3rd anchor point |
| anchor3Y | The y value of the 3rd anchor point |
| brush | The brush used to color draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx" \t "_blank)) |
| dashStyle | DashStyleHelper.Dash DashStyleHelper.DashDot DashStyleHelper.DashDotDot DashStyleHelper.Dot DashStyleHelper.Solid    **Note**: Drawing objects with y values very far off the visible canvas can lead to performance hits. Fancier DashStyles like DashDotDot will also require more resources than simple DashStyles like Solid. |
| width | The width of the draw object |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

**Examples**

| ns | |
| --- | --- |
| // Draws an Andrew's Pitchfork Draw.AndrewsPitchfork(this, "tag1", true, 4, Low[4], 3, High[3], 1, Low[1], Brushes.Blue, DashStyleHelper.Solid, 3); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.Arc()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/andrewspitchfork.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/arc.htm) |

**Definition**

Draws an arc.

**Method Return Value**

An [Arc](https://ninjatrader.com/es/support/helpGuides/nt8/arc.htm) object that represents the draw object.

**Syntax**

Draw.Arc(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush)  
Draw.Arc(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush)  
Draw.Arc(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, DashStyleHelper dashStyle, int width)  
Draw.Arc(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, DashStyleHelper dashStyle, int width)  
Draw.Arc(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.Arc(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.Arc(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, bool isGlobal, string templateName)  
Draw.Arc(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale. Default value is false. |
| startBarsAgo | The starting bar (x axis co-ordinate) where the draw object will be drawn. For example, a value of 10 would paint the draw object 10 bars back. |
| startTime | The starting time where the draw object will be drawn. |
| startY | The starting y value co-ordinate where the draw object will be drawn |
| endBarsAgo | The end bar (x axis co-ordinate) where the draw object will terminate |
| endTime | The end time where the draw object will terminate |
| endY | The end y value co-ordinate where the draw object will terminate |
| brush | The brush used to color draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx" \t "_blank)) |
| dashStyle | DashStyleHelper.Dash DashStyleHelper.DashDot DashStyleHelper.DashDotDot DashStyleHelper.Dot DashStyleHelper.Solid    **Note**: Drawing objects with y values very far off the visible canvas can lead to performance hits. Fancier DashStyles like DashDotDot will also require more resources than simple DashStyles like Solid. |
| width | The width of the draw object |
| drawOnPricePanel | Determines if the draw-object should be on the price panel or a separate panel |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

**Examples**

| ns |
| --- |
| // Draws a dotted lime green arc Draw.Arc(this, "tag1", false, 10, 1000, 0, 1001, Brushes.LimeGreen, DashStyleHelper.Dot, 2); |

|  |  |
| --- | --- |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.ArrowDown()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/arc.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/arrowdown.htm) |

**Definition**

Draws an arrow pointing down.

**Method Return Value**

An [ArrowDown](https://ninjatrader.com/es/support/helpGuides/nt8/arrowdown.htm) object that represents the draw object.

**Syntax**

Draw.ArrowDown(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush)  
Draw.ArrowDown(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush)  
Draw.ArrowDown(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush, bool drawOnPricePanel)  
Draw.ArrowDown(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush, bool drawOnPricePanel)  
Draw.ArrowDown(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, bool isGlobal, string templateName)  
Draw.ArrowDown(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale |
| barsAgo | The bar the object will be drawn at. A value of 10 would be 10 bars ago. |
| time | The time the object will be drawn at. |
| y | The y value |
| brush | The brush used to color draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx" \t "_blank)) |
| drawOnPricePanel | Determines if the draw-object should be on the price panel or a separate panel |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

|  |
| --- |
| **Tip**: The size of the arrow is tied to the chart's BarWidth and thus will scale automatically as the chart is resized |

**Examples**

| ns | |
| --- | --- |
| // Paints a red down arrow on the current bar 1 tick above the high   Draw.ArrowDown(this, "tag1", true, 0, High[0] + TickSize, Brushes.Red);   // Paints a blue down arrown on a three bar reversal pattern if (High[2] > High[3] && High[1] > High[2] && Close[0] < Open[0])     Draw.ArrowDown(this, CurrentBar.ToString(), true, 0, High[0] + TickSize, Brushes.Blue); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.ArrowLine()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/arrowdown.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/arrowline.htm) |

**Definition**

Draws an arrow line.

**Method Return Value**

An [ArrowLine](https://ninjatrader.com/es/support/helpGuides/nt8/arrowline.htm) object that represents the draw object.

**Syntax**

Draw.ArrowLine(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush)  
Draw.ArrowLine(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush)  
Draw.ArrowLine(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, DashStyleHelper dashStyle, int width)  
Draw.ArrowLine(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.ArrowLine(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.ArrowLine(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, bool isGlobal, string templateName)  
Draw.ArrowLine(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale. Default value is false. |
| startBarsAgo | The starting bar (x axis co-ordinate) where the draw object will be drawn. For example, a value of 10 would paint the draw object 10 bars back. |
| startTime | The starting time where the draw object will be drawn. |
| startY | The starting y value co-ordinate where the draw object will be drawn |
| endBarsAgo | The end bar (x axis co-ordinate) where the draw object will terminate |
| endTime | The end time where the draw object will terminate |
| endY | The end y value co-ordinate where the draw object will terminate |
| brush | The brush used to color draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx" \t "_blank)) |
| dashStyle | DashStyleHelper.Dash DashStyleHelper.DashDot DashStyleHelper.DashDotDot DashStyleHelper.Dot DashStyleHelper.Solid    **Note**: Drawing objects with y values very far off the visible canvas can lead to performance hits. Fancier DashStyles like DashDotDot will also require more resources than simple DashStyles like Solid. |
| width | The width of the draw object |
| drawOnPricePanel | Determines if the draw-object should be on the price panel or a separate panel |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

**Examples**

| ns | |
| --- | --- |
| // Draws a dotted lime green arrow line Draw.ArrowLine(this, "tag1", 10, 1000, 0, 1001, Brushes.LimeGreen, DashStyleHelper.Dot, 2); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.Diamond()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/arrowup.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/diamond.htm) |

**Definition**

Draws a diamond.

**Method Return Value**

A [Diamond](https://ninjatrader.com/es/support/helpGuides/nt8/diamond.htm) object that represents the draw object.

**Syntax**

Draw.Diamond(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush)  
Draw.Diamond(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush)  
Draw.Diamond(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush, bool drawOnPricePanel)  
Draw.Diamond(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush, bool drawOnPricePanel)  
Draw.Diamond(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, bool isGlobal, string templateName)  
Draw.Diamond(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale |
| barsAgo | The bar the object will be drawn at. A value of 10 would be 10 bars ago. |
| time | The time the object will be drawn at. |
| y | The y value |
| brush | The brush used to color draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx" \t "_blank)) |
| drawOnPricePanel | Determines if the draw-object should be on the price panel or a separate panel |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

|  |
| --- |
| **Tip**: The size of the diamond is tied to the chart's BarWidth and thus will scale automatically as the chart is resized |

**Examples**

| ns | |
| --- | --- |
| // Paints a red diamond on the current bar 1 tick below the low Draw.Diamond(this, "tag1", true, 0, Low[0] - TickSize, Brushes.Red); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.Dot()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/diamond.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/dot.htm) |

**Definition**

Draws a dot.

**Method Return Value**

A [Dot](https://ninjatrader.com/es/support/helpGuides/nt8/dot.htm) object that represents the draw object.

**Syntax**

Draw.Dot(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush)  
Draw.Dot(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush)  
Draw.Dot(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush, bool drawOnPricePanel)  
Draw.Dot(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush, bool drawOnPricePanel)  
Draw.Dot(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, bool isGlobal, string templateName)  
Draw.Dot(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale |
| barsAgo | The bar the object will be drawn at. A value of 10 would be 10 bars ago. |
| time | The time the object will be drawn at. |
| y | The y value |
| brush | The brush used to color draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx" \t "_blank)) |
| drawOnPricePanel | Determines if the draw-object should be on the price panel or a separate panel |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

|  |
| --- |
| **Tip**: The size of the dot is tied to the chart's BarWidth and thus will scale automatically as the chart is resized |

**Examples**

| ns | |
| --- | --- |
| // Paints a red dot on the current bar 1 tick below the low Draw.Dot(this, "tag1", true, 0, Low[0] - TickSize, Brushes.Red); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.Ellipse()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/dot.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/ellipse.htm) |

**Definition**

Draws an ellipse.

**Method Return Value**

An [Ellipse](https://ninjatrader.com/es/support/helpGuides/nt8/ellipse.htm) object that represents the draw object.

**Syntax**

Draw.Ellipse(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush)  
Draw.Ellipse(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, Brush areaBrush, int areaOpacity)  
Draw.Ellipse(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush)  
Draw.Ellipse(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, Brush areaBrush, int areaOpacity)  
Draw.Ellipse(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, bool drawOnPricePanel)  
Draw.Ellipse(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, Brush areaBrush, int areaOpacity, bool drawOnPricePanel)  
Draw.Ellipse(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, bool drawOnPricePanel)  
Draw.Ellipse(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, Brush areaBrush, int areaOpacity, bool drawOnPricePanel)  
Draw.Ellipse(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, bool isGlobal, string templateName)  
Draw.Ellipse(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale. Default value is false. |
| startBarsAgo | The starting bar (x axis co-ordinate) where the draw object will be drawn. For example, a value of 10 would paint the draw object 10 bars back |
| startTime | The starting time where the draw object will be drawn |
| startY | The starting y value co-ordinate where the draw object will be drawn |
| endBarsAgo | The end bar (x axis co-ordinate) where the draw object will terminate |
| endTime | The end time where the draw object will terminate |
| endY | The end y value co-ordinate where the draw object will terminate |
| brush | The brush used to color the outline of draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx" \t "_blank)) |
| areaBrush | The brush used to color the fill area of the draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx" \t "_blank)) |
| areaOpacity | Sets the level of transparency for the fill color. Valid values between 0 - 100. (0 = completely transparent, 100 = no opacity) |
| drawOnPricePanel | Determines if the draw-object should be on the price panel or a separate panel |
| isGlobalDrawingTool | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

**Examples**

| ns | |
| --- | --- |
| // Paints a red ellipse on the current bar Draw.Ellipse(this, "tag1", true, 5, Close[5], 0, Close[0], Brushes.Red, Brushes.Red, 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) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.ExtendedLine()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ellipse.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/extendedline.htm) |

**Definition**

Draws a line with infinite end points.

**Method Return Value**

An [ExtendedLine](https://ninjatrader.com/es/support/helpGuides/nt8/extendedline.htm) object that represents the draw object.

**Syntax**  
Draw.ExtendedLine(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush)  
Draw.ExtendedLine(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush)  
Draw.ExtendedLine(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, DashStyleHelper dashStyle, int width)  
Draw.ExtendedLine(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, DashStyleHelper dashStyle, int width)  
Draw.ExtendedLine(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.ExtendedLine(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.ExtendedLine(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, bool isGlobal, string templateName)  
Draw.ExtendedLine(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale. Default value is false. |
| startBarsAgo | The starting bar (x axis co-ordinate) where the draw object will be drawn. For example, a value of 10 would paint the draw object 10 bars back |
| startTime | The starting time where the draw object will be drawn |
| startY | The starting y value co-ordinate where the draw object will be drawn |
| endBarsAgo | The end bar (x axis co-ordinate) where the draw object will terminate |
| endTime | The end time where the draw object will terminate |
| endY | The end y value co-ordinate where the draw object will terminate |
| brush | The brush used to color draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx" \t "_blank)) |
| dashStyle | DashStyleHelper.Dash DashStyleHelper.DashDot DashStyleHelper.DashDotDot DashStyleHelper.Dot DashStyleHelper.Solid    **Note**: Drawing objects with y values very far off the visible canvas can lead to performance hits. Fancier DashStyles like DashDotDot will also require more resources than simple DashStyles like Solid. |
| width | The width of the draw object |
| drawOnPricePanel | Determines if the draw-object should be on the price panel or a separate panel |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

**Examples**

| ns | |
| --- | --- |
| // Draws a dotted lime green Draw.ExtendedLine(this, "tag1", 10, Close[10], 0, Close[0], Brushes.LimeGreen, DashStyleHelper.Dot, 2); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.FibonacciCircle()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/extendedline.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/fibonaccicircle.htm) |

**Definition**

Draws a fibonacci circle.

**Method Return Value**

A [FibonacciCircle](https://ninjatrader.com/es/support/helpGuides/nt8/fibonaccicircle.htm) object that represents the draw object.

**Syntax**  
Draw.FibonacciCircle(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY)  
Draw.FibonacciCircle(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY)  
Draw.FibonacciCircle(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, bool isGlobal, string templateName)  
Draw.FibonacciCircle(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale. Default value is false. |
| startBarsAgo | The starting bar (x axis co-ordinate) where the draw object will be drawn. For example, a value of 10 would paint the draw object 10 bars back. |
| startTime | The starting time where the draw object will be drawn |
| startY | The starting y value co-ordinate where the draw object will be drawn |
| endBarsAgo | The end bar (x axis co-ordinate) where the draw object will terminate |
| endTime | The end time where the draw object will terminate |
| endY | The end y value co-ordinate where the draw object will terminate |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

**Examples**

| ns | |
| --- | --- |
| // Draws a Fibonacci circle Draw.FibonacciCircle(this, "tag1", true, 10, Low[10], 0, High[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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.FibonacciExtensions()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/fibonaccicircle.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/fibonacciextensions.htm) |

**Definition**

Draws a fibonacci extension.

**Method Return Value**

A [FibonacciExtensions](https://ninjatrader.com/es/support/helpGuides/nt8/fibonacciextensions.htm) object that represents the draw object.

**Syntax**

Draw.FibonacciExtensions(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, int extensionBarsAgo, double extensionY)  
Draw.FibonacciExtensions(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, DateTime extensionTime, double extensionY)  
Draw.FibonacciExtensions(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, DateTime extensionTime, double extensionY, bool isGlobal, string templateName)  
Draw.FibonacciExtensions(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, int extensionBarsAgo, double extensionY, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale |
| startBarsAgo | The number of bars ago (x value) of the 1st anchor point |
| startTime | The time of the 1st anchor point |
| startY | The y value of the 1st anchor point |
| endBarsAgo | The number of bars ago (x value) of the 2nd anchor point |
| endTime | The time of the 2nd anchor point |
| endY | The y value of the 2nd anchor point |
| extensionBarsAgo | The number of bars ago (x value) of the 3rd anchor point |
| extensionTime | The time of the 3rd anchor point |
| extensionY | The y value of the 3rd anchor point |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

**Examples**

| ns | |
| --- | --- |
| // Draws a fibonnaci extension Draw.FibonacciExtensions(this, "tag1", true, 4, Low[4], 3, High[3], 1, Low[1]); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.FibonacciRetracements()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/fibonacciextensions.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/fibonacciretracements.htm) |

**Definition**

Draws a fibonacci retracement.

**Method Return Value**

A [FibonacciRetracements](https://ninjatrader.com/es/support/helpGuides/nt8/fibonacciretracements.htm) object that represents the draw object.

**Syntax**

Draw.FibonacciRetracements(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY)

Draw.FibonacciRetracements(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY)  
Draw.FibonacciRetracements(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, bool isGlobal, string templateName)  
Draw.FibonacciRetracements(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale. Default value is false. |
| startBarsAgo | The starting bar (x axis co-ordinate) where the draw object will be drawn. For example, a value of 10 would paint the draw object 10 bars back. |
| startTime | The starting time where the draw object will be drawn. |
| startY | The starting y value co-ordinate where the draw object will be drawn |
| endBarsAgo | The end bar (x axis co-ordinate) where the draw object will terminate |
| endTime | The end time where the draw object will terminate |
| endY | The end y value co-ordinate where the draw object will terminate |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

**Examples**

| ns | |
| --- | --- |
| // Draws a fibonnaci retracement Draw.FibonacciRetracements(this, "tag1", true, 10, Low[10], 0, High[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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **RemoveDrawObject()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/pricelevels.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/removedrawobjects.htm) |

**Definition**

Removes a draw object from the chart based on its tag value.

|  |
| --- |
| **Note**:  This method will **ONLY** remove DrawObjects which were created by a NinjaScript object.  User drawn objects **CANNOT** be removed from via NinjaScript |

**Method Return Value**

This method does not return a value

**Syntax**

RemoveDrawObject(string *tag*)

**Parameters**

|  |  |
| --- | --- |
| tag | A user defined unique id used to reference the draw object. For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |

**Examples**

| ns | |
| --- | --- |
| // Removes a draw object with the tag "tag1" RemoveDrawObject("tag1"); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **RemoveDrawObjects()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/removedrawobject.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/instruments_ninjascript.htm) |

**Definition**

Removes all draw objects originating from the indicator or strategy from the chart.

|  |
| --- |
| **Note**:  This method will **ONLY** remove DrawObjects which were created by a NinjaScript object.  User drawn objects **CANNOT** be removed from via NinjaScript |

**Method Return Value**

This method does not return a value

**Syntax**

RemoveDrawObjects()

**Examples**

| ns | |
| --- | --- |
| // Removes all draw objects RemoveDrawObjects(); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.Text()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/square.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/text.htm) |

**Definition**

Draws text.

**Method Return Value**

A [Text](https://ninjatrader.com/es/support/helpGuides/nt8/text.htm) object that represents the draw object.

**Syntax**

Draw.Text(NinjaScriptBase owner, string tag, string text, int barsAgo, double y)  
Draw.Text(NinjaScriptBase owner, string tag, string text, int barsAgo, double y, Brush textBrush)  
Draw.Text(NinjaScriptBase owner, string tag, string text, int barsAgo, double y, bool isGlobal, string templateName)  
Draw.Text(NinjaScriptBase owner, string tag, bool isAutoScale, string text, int barsAgo, double y, int yPixelOffset, Brush textBrush, SimpleFont font, TextAlignment alignment, Brush outlineBrush, Brush areaBrush, int areaOpacity)  
Draw.Text(NinjaScriptBase owner, string tag, bool isAutoScale, string text, DateTime time, double y, int yPixelOffset, Brush textBrush, SimpleFont font, TextAlignment alignment, Brush outlineBrush, Brush areaBrush, int areaOpacity)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale. Default value is false. |
| text | The text you wish to draw |
| barsAgo | The bar (x axis co-ordinate) where the draw object will be drawn. For example, a value of 10 would paint the draw object 10 bars back. |
| time | The time where the draw object will be drawn. |
| y | The y co-ordinate location the object will be drawn |
| yPixelOffset | The offset value in pixels from within the text box area |
| textBrush | The brush used to color the text of the draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.textalignment%28v=vs.110%29.aspx" \t "_blank)) |
| font | A [Simple Font](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm) object |
| alignment | TextAlignment.Center,  TextAlignment.Left,  TextAlignment.Right,  TextAlignment.Justify ([reference](https://msdn.microsoft.com/en-us/library/system.windows.textalignment(v=vs.110).aspx" \t "_blank)) |
| outlineBrush | The brush used to color the text box outline ([reference](http://msdn.microsoft.com/en-us/library/system.drawing.color_members(v=vs.90).aspx" \t "_blank)) |
| areaBrush | The brush used to color the text box fill area ([reference](http://msdn.microsoft.com/en-us/library/system.drawing.color_members(v=vs.90).aspx" \t "_blank)) |
| areaOpacity | Sets the level of transparency for the fill color. Valid values between 0 - 100. (0 = completely transparent, 100 = no opacity) |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

**Examples**

| ns |
| --- |
| // Draws text Draw.Text(this, "tag1", "Text to draw", 10, 1000, Brushes.Black); |

|  |
| --- |
| **Tip**:  In some cases, it may be useful to pass in the [ChartControl.Properties](https://ninjatrader.com/es/support/helpGuides/nt8/chartcontrol_properties.htm) **TextFont** brush as well as the **LabelFont** [SimpleFont](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm) object to render your custom text .  This will help ensure that the text will be visible and match what a user has configured for their chart label display settings. |

| ns | |
| --- | --- |
| // match the text brush to what the user has configured on their chart Draw.Text(this, "tag1", "Text to draw", 10, 1000, ChartControl.Properties.ChartText); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.Dot()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/diamond.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/dot.htm) |

**Definition**

Draws a dot.

**Method Return Value**

A [Dot](https://ninjatrader.com/es/support/helpGuides/nt8/dot.htm) object that represents the draw object.

**Syntax**

Draw.Dot(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush)  
Draw.Dot(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush)  
Draw.Dot(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush, bool drawOnPricePanel)  
Draw.Dot(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush, bool drawOnPricePanel)  
Draw.Dot(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, bool isGlobal, string templateName)  
Draw.Dot(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale |
| barsAgo | The bar the object will be drawn at. A value of 10 would be 10 bars ago. |
| time | The time the object will be drawn at. |
| y | The y value |
| brush | The brush used to color draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx" \t "_blank)) |
| drawOnPricePanel | Determines if the draw-object should be on the price panel or a separate panel |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

|  |
| --- |
| **Tip**: The size of the dot is tied to the chart's BarWidth and thus will scale automatically as the chart is resized |

**Examples**

| ns | |
| --- | --- |
| // Paints a red dot on the current bar 1 tick below the low Draw.Dot(this, "tag1", true, 0, Low[0] - TickSize, Brushes.Red); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) > [Draw.HorizontalLine()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_horizontalline.htm) >  **HorizontalLine** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_horizontalline.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_horizontalline.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_line.htm) |

**Definition**

Represents an interface that exposes information regarding a Horizontal Line [IDrawingTool.](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm)

**Methods and Properties**

|  |  |
| --- | --- |
| StartAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm" \l "chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm" \l "chartanchor) representing the end point of the drawing object |
| Stroke | A [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the object |

**Example**

| ns | |
| --- | --- |
| // Instantiate a HorizontalLine object HorizontalLine myLine = Draw.HorizontalLine(this, "tag1", 1000, Brushes.Black);   // Set a new Stroke for the object myLine.Stroke = new Stroke(Brushes.Green, DashStyleHelper.Dash, 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) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.HorizontalLine()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/gannfan.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/horizontalline.htm) |

**Definition**

Draws a horizontal line.

**Method Return Value**

A [HorizontalLine](https://ninjatrader.com/es/support/helpGuides/nt8/horizontalline.htm) object that represents the draw object.

**Syntax**

Draw.HorizontalLine(NinjaScriptBase owner, string tag, double y, Brush brush)  
Draw.HorizontalLine(NinjaScriptBase owner, string tag, bool isAutoScale, double y, Brush brush, DashStyleHelper dashStyle, int width)  
Draw.HorizontalLine(NinjaScriptBase owner, string tag, bool isAutoscale, double y, Brush brush, bool drawOnPricePanel)  
Draw.HorizontalLine(NinjaScriptBase owner, string tag, double y, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.HorizontalLine(NinjaScriptBase owner, string tag, double y, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| isAutoScale | Determines if the draw object will be included in the y-axis scale. Default value is false. |
| y | The y value |
| brush | The brush used to color draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx" \t "_blank)) |
| dashStyle | DashStyleHelper.Dash DashStyleHelper.DashDot DashStyleHelper.DashDotDot DashStyleHelper.Dot DashStyleHelper.Solid    **Note**: Fancier DashStyles like DashDotDot will require more resources than simple DashStyles like Solid. |
| width | The width of the draw object |
| isDrawOnPricePanel | Determines if the draw-object should be on the price panel or a separate panel |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

**Examples**

| ns | |
| --- | --- |
|  | // Draws a horizontal line Draw.HorizontalLine(this, "tag1", 1000, Brushes.Black); |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.VerticalLine()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/triangleup.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/verticalline.htm) |

**Definition**

Draws a vertical line.

**Method Return Value**

A [VerticalLine](https://ninjatrader.com/es/support/helpGuides/nt8/verticalline.htm) object that represents the draw object.

**Syntax**

Draw.VerticalLine(NinjaScriptBase owner, string tag, DateTime time, Brush brush)  
Draw.VerticalLine(NinjaScriptBase owner, string tag, DateTime time, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.VerticalLine(NinjaScriptBase owner, string tag, int barsAgo, Brush brush)  
Draw.VerticalLine(NinjaScriptBase owner, string tag, int barsAgo, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.VerticalLine(NinjaScriptBase owner, string tag, int barsAgo, bool isGlobal, string templateName)  
Draw.VerticalLine(NinjaScriptBase owner, string tag, DateTime time, bool isGlobal, string templateName)

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| barsAgo | The bar the object will be drawn at. A value of 10 would be 10 bars ago. |
| time | The time the object will be drawn at. |
| brush | The brush used to color draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx" \t "_blank)) |
| dashStyle | DashStyleHelper.Dash DashStyleHelper.DashDot DashStyleHelper.DashDotDot DashStyleHelper.Dot DashStyleHelper.Solid    **Note**: Fancier DashStyles like DashDotDot will require more resources than simple DashStyles like Solid. |
| width | The width of the draw object |
| drawOnPricePanel | Determines if the draw-object should be on the price panel or a separate panel |
| isGlobal | Determines if the draw object will be global across all charts which match the instrument |
| templateName | The name of the drawing tool template the object will use to determine various visual properties (empty string could be used to just use the UI default visuals instead) |

**Examples**

| ns | |
| --- | --- |
| // Draws a vertical line Draw.VerticalLine(this, "tag1", 10, Brushes.Black); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) > [Draw.VerticalLine()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_verticalline.htm) >  **VerticalLine** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_verticalline.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_verticalline.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/brushes.htm) |

**Definition**

Represents an interface that exposes information regarding a Vertical Line [IDrawingTool](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm).

**Methods and Properties**

|  |  |
| --- | --- |
| StartAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm" \l "chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm" \l "chartanchor) representing the end point of the drawing object |
| Stroke | A [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the object |

**Examples**

| ns | |
| --- | --- |
| // Instantiate a VerticalLine object VerticalLine myLine = Draw.VerticalLine(this, "tag1", 10, Brushes.Black);   // Change the object's Stroke myLine.Stroke = new Stroke(Brushes.BlanchedAlmond, DashStyleHelper.Dot, 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) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **DrawObjects** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/candleoutlinebrushes.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm) |

**Definition**

A collection holding all of the drawn chart objects on the chart, for all series. The draw objects can be manually drawn or script generated objects.

|  |
| --- |
| **Notes**:    •When reloading NinjaScript, all objects (including manual drawing tools) are reloaded at the same time. There is no guarantee a manually drawn object will be added to the **DrawObjects** collection before an indicator starts processing data.  •DrawObjects.ToList() is thread safe. DrawObjects collection itself is still dynamic (meaning it updates live) and as a result you can still run the risk of the collection being modified while you try to read it (and thus would see the related C# log entry) However, DrawObjects.ToList() is a snapshot of DrawObjects collection at the time the call is made.  •Also please keep in mind that iterating over a large DrawObjects collection could have an impact on performance  •Draw objects are disposed (for example on chart closing) after State.Terminated is seen for your custom NinjaScript studies potentially working with those |

**Property Value**

A collection of [IDrawingTool](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm) objects.

**Syntax**

DrawObjects  
DrawObjects[string tag]  
DrawObjects.Count

**Examples**

| ns**Finding the draw object of a specific tag** |
| --- |
| protected override void OnBarUpdate() {   if (DrawObjects["someTag"] != null && DrawObjects["someTag"] is DrawingTools.Line)   {     // Do something with the drawing tool line   }             // An alternative approach to find the draw object by a tag   if (DrawObjects["someTag"] as DrawingTools.Line != null)   {     // Do something drawing tool line   }       // Yet another way to find a drawing tool by a tag   if (DrawObjects["someTag"].GetType().Name == "Line")   {     // Do something drawing tool line     } } |

| ns**Get the number of draw objects on a chart** |
| --- |
| protected override void OnBarUpdate() {   if (DrawObjects.Count == 3)   {         // Do something   } } |

| ns**Looping through the collection to find specific draw objects** |
| --- |
| protected override void OnBarUpdate() {   // Loops through the DrawObjects collection via a threadsafe list copy   foreach (DrawingTool draw in DrawObjects.ToList())   {     // Finds line objects that are attached globally to all charts of the same instrument     if (draw.IsGlobalDrawingTool && draw is DrawingTools.Line)     {         DrawingTools.Line globalLine = draw as DrawingTools.Line;                                 // Changes the line color and prints its starting and end points         globalLine.Stroke.Brush = Brushes.Black;          Print("Start: " + globalLine.StartAnchor.SlotIndex + " End: " + globalLine.EndAnchor.SlotIndex);     }       // Finds non-global line objects     else if (draw is DrawingTools.Line)     {                       // Indicates if this is a manually drawn or script generated line         Print("Line Object: " + draw.Tag + " Manually Drawn: " + draw.IsUserDrawn);     }   }   } |

|  |  |
| --- | --- |
| **Note**: Typecasting as in the example above will not function the same way in a compiled assembly (DLL). For an alternative approach, see the [Considerations For Compiled Assemblies](https://ninjatrader.com/es/support/helpGuides/nt8/considerations_for_compiled_assemblies.htm) page. | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **DrawOnPricePanel** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawhorizontalgridlines.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawverticalgridlines.htm) |

**Definition**

Determines the chart panel the draw objects renders

**Property Value**

This property returns **true** if the indicator paints draw objects on the price panel; otherwise when false, draw objects are painted on the actual indicator panel itself. Default set to **true**.

|  |
| --- |
| **Warning**:  This property should **ONLY** be set from the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method during **State.SetDefaults.**Dynamically using DrawOnPricePanel in an indicator outside of State.SetDefaults may show issues when working with that indicator through a hosting strategy via [AddChartIndicator()](https://ninjatrader.com/es/support/helpGuides/nt8/addchartindicator.htm). |

**Syntax**

DrawOnPricePanel

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)      {           DrawOnPricePanel = false; // Draw objects now paint on the indicator panel itself           AddPlot(Brushes.Orange, "SMA");      } } | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **AllowRemovalOfDrawObjects** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/brushes.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/backbrush.htm) |

**Definition**

Determines if programmatically drawn [DrawObjects](https://ninjatrader.com/es/support/helpGuides/nt8/drawingtools_drawobjects.htm) are allowed to remove manually from the chart

**Property Value**

When set to **true**, the draw objects from the indicator or strategy can be deleted from the chart manually by a user. If **false**, draw objects from the indicator or strategy can only be removed from the chart if the script removes the drawing object, or the script is terminates.  Default set to **false**.

**Syntax**

AllowRemovalOfDrawObjects

**Examples**

|  |  |
| --- | --- |
| ns |  |
| protected override void OnStateChange() {     Add(new Plot(Brushes.Orange, "SMA"));     AllowRemovalOfDrawObjects = true; // Draw objects can be removed separately from the script } | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **RemoveDrawObject()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/pricelevels.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/removedrawobjects.htm) |

**Definition**

Removes a draw object from the chart based on its tag value.

|  |
| --- |
| **Note**:  This method will **ONLY** remove DrawObjects which were created by a NinjaScript object.  User drawn objects **CANNOT** be removed from via NinjaScript |

**Method Return Value**

This method does not return a value

**Syntax**

RemoveDrawObject(string *tag*)

**Parameters**

|  |  |
| --- | --- |
| tag | A user defined unique id used to reference the draw object. For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |

**Examples**

| ns | |
| --- | --- |
| // Removes a draw object with the tag "tag1" RemoveDrawObject("tag1"); | |
| **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) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **RemoveDrawObjects()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/removedrawobject.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/instruments_ninjascript.htm) |

**Definition**

Removes all draw objects originating from the indicator or strategy from the chart.

|  |
| --- |
| **Note**:  This method will **ONLY** remove DrawObjects which were created by a NinjaScript object.  User drawn objects **CANNOT** be removed from via NinjaScript |

**Method Return Value**

This method does not return a value

**Syntax**

RemoveDrawObjects()

**Examples**

| ns | |
| --- | --- |
| // Removes all draw objects RemoveDrawObjects(); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) >  **Indicator** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onnextdatapoint.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/addline.htm) |

The methods and properties covered in this section are unique to custom indicator development.  Indicator configuration properties globally define various behaviors of indicators. All properties have default values and can be overridden by setting them in the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method of the indicator.

|  |
| --- |
| **Tip**:  See also the "[Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm)" section for more method and properties which are shared by NinjaScript types |

**Methods and Properties**

|  |  |
| --- | --- |
| [AddLine()](https://ninjatrader.com/es/support/helpGuides/nt8/addline.htm) | Adds line objects on a chart. |
| [AddPlot()](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm) | Adds plot objects that define how an indicator or strategy data series render on a chart. |
| [BarsRequiredToPlot](https://ninjatrader.com/es/support/helpGuides/nt8/barsrequiredtoplot.htm) | The number of bars on a chart required before the script plots. |
| [DisplayInDataBox](https://ninjatrader.com/es/support/helpGuides/nt8/displayindatabox.htm) | Determines if plot(s) display in the chart data box. |
| [DrawHorizontalGridLines](https://ninjatrader.com/es/support/helpGuides/nt8/drawhorizontalgridlines.htm) | Plots horizontal grid lines on the indicator panel. |
| [DrawOnPricePanel](https://ninjatrader.com/es/support/helpGuides/nt8/drawonpricepanel.htm) | Determines the chart panel the draw objects renders. |
| [DrawVerticalGridLines](https://ninjatrader.com/es/support/helpGuides/nt8/drawverticalgridlines.htm) | Plots vertical grid lines on the indicator panel. |
| [IndicatorBaseConverter](https://ninjatrader.com/es/support/helpGuides/nt8/indicatorbaseconverter.htm) | A custom TypeConverter class handling the designed behavior of an indicator's property descriptor collection. |
| [IsChartOnly](https://ninjatrader.com/es/support/helpGuides/nt8/ischartonly.htm) | If true, any indicator will be only available for charting usage - indicators with this property enabled would for example not be expected to show if called in a SuperDOM or MarketAnalyzer window. |
| [IsSuspendedWhileInactive](https://ninjatrader.com/es/support/helpGuides/nt8/issuspendedwhileinactive.htm) | Prevents real-time market data events from being raised while the indicator's hosting feature is in a state that would be considered suspended and not in immediate use by a user. |
| [PaintPriceMarkers](https://ninjatrader.com/es/support/helpGuides/nt8/paintpricemarkers.htm) | If true, any indicator plot values display price markers in the y-axis. |
| [ShowTransparentPlotsInDataBox](https://ninjatrader.com/es/support/helpGuides/nt8/showtransparentplotsindatabox.htm) | Determines if plot(s) values which are set to a Transparent brush display in the chart data box. |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **AddLine()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/arelinesconfigurable.htm) |

**Definition**

Adds line objects on a chart.

|  |
| --- |
| **Note:**  Lines are **ONLY** visible from the UI property grid when AddLine() is called from **State.SetDefaults**. If your indicator or strategy dynamically adds lines during **State.Configure**, you will **NOT** have an opportunity to select the line or to set the line configuration via the UI. Alternatively, you may use custom public [Brush](https://ninjatrader.com/es/support/helpGuides/nt8/brushes.htm), [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) or value properties which are accessible in the **State.SetDefaults** and pass those values to AddLine() during**State.Configure**. Calling AddLine() in this manner should be reserved for special cases.  Please see the examples below. |

**Methods and Properties**

|  |  |
| --- | --- |
| [AreLinesConfigurable](https://ninjatrader.com/es/support/helpGuides/nt8/arelinesconfigurable.htm) | Determines if the [line](https://ninjatrader.com/es/support/helpGuides/nt8/addline.htm)(s) used in an indicator are configurable from within the indicator dialog window. |
| [Line Class](https://ninjatrader.com/es/support/helpGuides/nt8/line_class.htm) | Objects derived from the Line class are used to characterize how an oscillator line is visually displayed (plotted) on a chart. |
| [Lines](https://ninjatrader.com/es/support/helpGuides/nt8/lines.htm) | A collection holding all of the Line objects that define the visualization characteristics oscillator lines of the indicator. |

**Syntax**

AddLine(Brush brush, double value, string name)  
AddLine(Stroke stroke, double value, string name)

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

**Parameters**

|  |  |
| --- | --- |
| brush | A Brush object used to construct the line |
| name | A string value representing the name of the line |
| stroke | A Stroke object used to construct the line |
| value | A double value representing the value the line will be drawn at |

**Examples**

| ns | **Defining a single UI configurable static line** |
| --- | --- |
|  | protected override void OnStateChange() {         if (State == State.SetDefaults)   {     Name = "Examples Indicator";       // Adds an oscillator line at a value of 30     AddLine(Brushes.Gray, 30, "Lower");   } } |

| ns | **Indicator which dynamically adds a line in State.Configure** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name                 = "Examples Indicator";       // logical property which user can set     UseSpecialMode   = false;     // Default brush selection pushed to the UI     MyBrush = Brushes.Red;   }   else if (State == State.Configure)   {     // if user enables logical property     if (UseSpecialMode)     {         // add line using default selected brush and special line name         AddLine(MyBrush, 40, "My Special Line");     }     else     {         // otherwise use default selected brush and regular line name         AddLine(MyBrush, 60, "My Regular Line");     }   } }     [XmlIgnore] public Brush MyBrush { get; set; }   public bool UseSpecialMode { 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) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **AddPlot()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/lines.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/areplotsconfigurable.htm) |

**Definition**

Adds plot objects that define how an indicator or strategy data series render on a chart. When this method is called to add a plot, an associated [Series<double>](https://ninjatrader.com/es/support/helpGuides/nt8/seriest.htm) object is created held in the [Values](https://ninjatrader.com/es/support/helpGuides/nt8/value.htm) collection.

|  |
| --- |
| **Note:**  Plots are **ONLY** visible from the UI property grid when AddPlot() is called from **State.SetDefaults**. If your indicator or strategy dynamically adds plots during **State.Configure**, you will **NOT** have an opportunity to select the plot or to set the plot configuration via the UI.  Alternatively, you may use custom public [Brush](https://ninjatrader.com/es/support/helpGuides/nt8/brushes.htm), [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm), or **PlotStyle** properties which are accessible in **State.SetDefaults** and pass those values to AddPlot() during**State.Configure**.  Calling AddPlot() in this manner should be reserved for special cases.  Please see the examples below. |

**Methods and Properties**

|  |  |
| --- | --- |
| [ArePlotsConfigurable](https://ninjatrader.com/es/support/helpGuides/nt8/areplotsconfigurable.htm) | Determines if the plot(s) used in an indicator are configurable within the indicator dialog window. |
| [Displacement](https://ninjatrader.com/es/support/helpGuides/nt8/displacement.htm) | An offset value that shifts the visually displayed value of an indicator. |
| [PlotBrushes](https://ninjatrader.com/es/support/helpGuides/nt8/plotbrushes.htm) | Holds an array of color series objects holding historical bar colors. |
| [Plots](https://ninjatrader.com/es/support/helpGuides/nt8/plots.htm) | A collection holding all of the Plot objects that define their visualization characteristics. |

**Syntax**

AddPlot(Brush brush, string name)  
AddPlot(Stroke stroke, PlotStyle plotStyle, string name)

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

**Parameters**

|  |  |
| --- | --- |
| brush | A Brush object used to construct the plot |
| name | A string representing the name of the plot |
| plotStyle | A PlotStyle object used to construct the style of the plot    Possible values:   PlotStyle.Bar PlotStyle.Block PlotStyle.Cross PlotStyle.Dot PlotStyle.Hash PlotStyle.HLine PlotStyle.Line  PlotStyle.PriceBox PlotStyle.Square PlotStyle.TriangleDown PlotStyle.TriangleLeft PlotStyle.TriangleRight PlotStyle.TriangleUp |
| stroke | A Stroke object used to construct the plot |

|  |
| --- |
| **Tips:**  1.We suggest using the NinjaScript wizard to generate your plots.  2.[Plot](https://ninjatrader.com/es/support/helpGuides/nt8/plots.htm) objects **DO NOT** hold the actual script values. They simply define how the script's values are plotted on a chart.  3.A script may calculate multiple values and therefore hold multiple plots to determine the display of each calculated value. Script values are held in the script's [Values](https://ninjatrader.com/es/support/helpGuides/nt8/value.htm) collection.  4.If you script calls AddPlot() multiple times, then multiple Values series are added per the "three value series" example below  5.For [MultiSeries scripts](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm), plots are synched to the primary series of the NinjaScript object. |

**Examples**

| ns | **Indicator using various AddPlot() signatures** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "Examples Indicator";       // Adds a blue line style plot     AddPlot(Brushes.Blue, "myPlot");       // Adds a blue historgram style plot     AddPlot(new Stroke(Brushes.Blue), PlotStyle.Bar, "myPlot");   } } |

| ns | **Indicator which adds three value series** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "Examples Indicator";       // Add three plots and associated Series<double> objects     AddPlot(Brushes.Blue, "PlotA");     // Defines the plot for Values[0]     AddPlot(Brushes.Red, "PlotB");     // Defines the plot for Values[1]     AddPlot(Brushes.Green, "PlotC");   // Defines the plot for Values[2]   } } protected override void OnBarUpdate() {   Values[0][0] = Median[0];   // Blue "Plot A"   Values[1][0] = Low[0];       // Red "Plot B"   Values[2][0] = High[0];     // Green "Plot C" } |

| ns | **Indicator which dynamically adds a plot in State.Configure** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name                 = "Examples Indicator";       // logical property which user can set     UseSpecialMode   = false;     // Default brush selection pushed to the UI     MyBrush = Brushes.Red;   }   else if (State == State.Configure)   {     // if user enables logical property     if (UseSpecialMode)     {         // add plot using default selected brush and special plot name         AddPlot(MyBrush, "My Special Plot");     }     else     {         // otherwise use default selected brush and regular plot name         AddPlot(MyBrush, "My Regular Plot");     }   } }   protected override void OnBarUpdate() {   if (UseSpecialMode)     Value[0] = Close[0] + High[0] / 2;     else Value[0] = Close[0] \* TickSize / 2; }   [XmlIgnore] public Brush MyBrush { get; set; }   public bool UseSpecialMode { 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) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **BarsRequiredToPlot** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/plots.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/displayindatabox.htm) |

**Definition**

The number of bars on a chart required before the script plots.

|  |
| --- |
| **Note**:  This property is **NOT** the same as a minimum number of bars required to calculate the script values.  OnBarUpdate will always start calculating for the first bar on the chart (CurrentBar 0) |

**Property Value**

An int value that represents the minimum number of bars required.

**Syntax**

BarsRequiredToPlot

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {      if (State == State.SetDefaults)      {           BarsRequiredToPlot = 10; // Do not plot until the 11th bar on the chart          AddPlot(Brushes.Orange, "SMA");      }     } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **DisplayInDataBox** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/barsrequiredtoplot.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawhorizontalgridlines.htm) |

**Definition**

Determines if plot(s) display in the chart data box.

**Property Value**

This property returns **true** if the indicator plot(s) values display in the chart data box; 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**

DisplayInDataBox

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         DisplayInDataBox = false;           AddPlot(Brushes.Orange, "SMA");     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **DrawHorizontalGridLines** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/displayindatabox.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawonpricepanel.htm) |

**Definition**

Plots horizontal grid lines on the indicator panel.

|  |
| --- |
| **Note**:  The indicator panel's parent chart has a similar option 'Grid line - horizontal  which if Visible property set to **false**, will override the indicator's local setting if **true**. |

**Property Value**

This property returns **true** if horizontal grid lines are plotted on the indicator panel; otherwise, **false**. Default set to **true**.

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

**Syntax**

DrawHorizontalGridLines

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         DrawHorizontalGridLines = false; // Horizontal grid lines will not plot on the indicator panel             AddPlot(Brushes.Orange, "SMA");     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **DrawOnPricePanel** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawhorizontalgridlines.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawverticalgridlines.htm) |

**Definition**

Determines the chart panel the draw objects renders

**Property Value**

This property returns **true** if the indicator paints draw objects on the price panel; otherwise when false, draw objects are painted on the actual indicator panel itself. Default set to **true**.

|  |
| --- |
| **Warning**:  This property should **ONLY** be set from the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method during **State.SetDefaults.**Dynamically using DrawOnPricePanel in an indicator outside of State.SetDefaults may show issues when working with that indicator through a hosting strategy via [AddChartIndicator()](https://ninjatrader.com/es/support/helpGuides/nt8/addchartindicator.htm). |

**Syntax**

DrawOnPricePanel

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)      {           DrawOnPricePanel = false; // Draw objects now paint on the indicator panel itself           AddPlot(Brushes.Orange, "SMA");      } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **DrawOnPricePanel** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawhorizontalgridlines.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawverticalgridlines.htm) |

**Definition**

Determines the chart panel the draw objects renders

**Property Value**

This property returns **true** if the indicator paints draw objects on the price panel; otherwise when false, draw objects are painted on the actual indicator panel itself. Default set to **true**.

|  |
| --- |
| **Warning**:  This property should **ONLY** be set from the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method during **State.SetDefaults.**Dynamically using DrawOnPricePanel in an indicator outside of State.SetDefaults may show issues when working with that indicator through a hosting strategy via [AddChartIndicator()](https://ninjatrader.com/es/support/helpGuides/nt8/addchartindicator.htm). |

**Syntax**

DrawOnPricePanel

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)      {           DrawOnPricePanel = false; // Draw objects now paint on the indicator panel itself           AddPlot(Brushes.Orange, "SMA");      } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **IndicatorBaseConverter Class** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawverticalgridlines.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/ischartonly.htm) |

**Definition**

A custom [TypeConverter](https://msdn.microsoft.com/en-us/library/system.componentmodel.typeconverter%28v=vs.110%29.aspx" \t "_blank) class handling the designed behavior of an indicator's property descriptor collection.  Use this as a base class for any custom **TypeConverter** you are applying to an indicator class.

|  |
| --- |
| **Notes:**  •A working NinjaScript demo can be found through the reference sample on "[Using a TypeConverter to Customize Property Grid Behavior](http://ninjatrader.com/support/forum/showthread.php?t=97919" \t "_blank)"  •When applying the custom converter, you must fully qualify the name (e.g., "NinjaTrader.NinjaScript.Indicators.MyCustomConveter")  •Additional **TypeConverter** information can be found from the [MSDN documentation](https://msdn.microsoft.com/en-us/library/system.componentmodel.typeconverter%28v=vs.110%29.aspx)  •See also [TypeConverterAttribute](https://ninjatrader.com/es/support/helpGuides/nt8/typeconverterattribute.htm)  •For Strategies, see the [StrategyBaseConverter](https://ninjatrader.com/es/support/helpGuides/nt8/strategybaseconverter.htm) class |

**Relevant base methods**

|  |  |
| --- | --- |
| [TypeConverter.GetProperties()](https://msdn.microsoft.com/en-us/library/system.componentmodel.typeconverter.getproperties(v=vs.110).aspx) | When overriding **GetProperties()**, calling base.GetProperties() ensures that all default property grid behavior works as designed |
| [TypeConverter.GetPropertiesSupported()](https://msdn.microsoft.com/en-us/library/system.componentmodel.typeconverter.getpropertiessupported(v=vs.110).aspx) | In your custom converter class, you must override **GetPropertiesSupported()**and return a value of **true** in order for your custom type converter to work |

**Syntax**

public class IndicatorBaseConverter : TypeConverter

|  |
| --- |
| **Warning**:  Failure to apply a type of **IndicatorBaseConverter** on an indicator class can result in unpredictable behavior of the standard NinjaTrader WPF property grid. |

|  |
| --- |
| **Tip**: Common indicator functions like Print() are not available to a type converter instance.  To debug a type converter class, you can use the AddOn [Debug Concepts](https://ninjatrader.com/es/support/helpGuides/nt8/alert_and_debug_concepts.htm) or [attach to a debugger](https://ninjatrader.com/es/support/helpGuides/nt8/visual_studio_debugging.htm) (recommended) |

**Examples**

| ns | |
| --- | --- |
| //This namespace holds Indicators in this folder and is required. Do not change it. namespace NinjaTrader.NinjaScript.Indicators {   // When applying the type converter, you must fully qualify the name   [TypeConverter("NinjaTrader.NinjaScript.Indicators.MyCustomConveter")]   public class MyCustomIndicator : Indicator   {     protected override void OnStateChange()     {         if (State == State.SetDefaults)         {           Name   = "MyCustomIndicator";         }     }       protected override void OnBarUpdate()     {         //Add your custom indicator logic here.     }   }     public class MyCustomConveter : IndicatorBaseConverter   {     // A general TypeConveter method used for converting types     public override PropertyDescriptorCollection GetProperties(ITypeDescriptorContext context, object component, Attribute[] attrs)     {         // sometimes you may need the indicator instance which actually exists on the grid         MyCustomIndicator indicator = component as MyCustomIndicator;           // base.GetProperties ensures we have all the properties (and associated property grid editors)         // NinjaTrader internal logic handles for a given indicator         PropertyDescriptorCollection propertyDescriptorCollection = base.GetPropertiesSupported(context)                 ? base.GetProperties(context, component, attrs) : TypeDescriptor.GetProperties(component, attrs);           if (indicator == null || propertyDescriptorCollection == null)           return propertyDescriptorCollection;           // example of why you may need the instance that exists on the grid....         if (indicator.EntryHandling == EntryHandling.UniqueEntries)         {           // do something in the event a property contains some value...         }           // Loop all of the properties of the indicator         foreach (PropertyDescriptor property in propertyDescriptorCollection)         {           // do something with a specific property             // cannot call Print() here           // but you can call the static Output window "Process()"           NinjaTrader.Code.Output.Process(property.Name, PrintTo.OutputTab1);         }           // must return the collection after making changes         return propertyDescriptorCollection;     }       // Important:  This must return true otherwise the type converter will not be called     public override bool GetPropertiesSupported(ITypeDescriptorContext context)     { return true; }     }   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **IsChartOnly** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/indicatorbaseconverter.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/issuspendedwhileinactive.htm) |

**Definition**

If true, any indicator will be only available for charting usage - indicators with this property enabled would for example not be expected to show if called in a SuperDOM or MarketAnalyzer window.

**Property Value**

This property returns **true** if the indicator can only be used on a chart; otherwise, **false**. Default set to **false**.

|  |
| --- |
| **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**

IsChartOnly

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         IsChartOnly = true; // Allow the indicator to work in charting environment only           } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **IsSuspendedWhileInactive** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ischartonly.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/paintpricemarkers.htm) |

**Definition**

Prevents OnBarUpdate from being raised while the indicators display is not in use.  Enabling this property in your indicator helps save CPU cycles while the indicator is suspended and not in use by a user.  Once the indicator is in a state that would no longer be considered suspended, the historical OnBarUpdate() events will be triggered allowing the indicator to catch up to current real-time values.

Suspension occurs in the following scenarios:

•Minimized Chart

•Minimized Market Analyzer

•Minimized Hot List Analyzer

•Minimized SuperDOM

•Background tabs of above features are considered "minimized"

•Inactive workspaces in the background

|  |
| --- |
| **Note**:  Since events in OnBarUpdate() will not be processed while the indicator is suspended, internal NinjaScript functions such as [Alert()](https://ninjatrader.com/es/support/helpGuides/nt8/alert.htm), [PlaySound()](https://ninjatrader.com/es/support/helpGuides/nt8/playsound.htm), [Share()](https://ninjatrader.com/es/support/helpGuides/nt8/share.htm), [Print()](https://ninjatrader.com/es/support/helpGuides/nt8/print.htm), etc - or any other method that would be used to notify a user of activity will **NOT** be processed until the indicator is un-suspended. |

**Scenarios where suspension will not occur**

The **IsSuspendedWhileInactive** property will be ignored and real-time events will be processed as normal under the following cases:

•Indicators running in [Automated NinjaScript Strategies](https://ninjatrader.com/es/support/helpGuides/nt8/running_a_ninjascript_strategy.htm)

•Indicators which have [manually configured alerts](https://ninjatrader.com/es/support/helpGuides/nt8/alerts_dialog.htm)

•Indicators which have been [manually attached to orders](https://ninjatrader.com/es/support/helpGuides/nt8/attachingorderstoindicators.htm)

**Property Value**

This property returns **true** if indicator can take advantage of suspension optimization; otherwise, **false**. Default set to **false**.

|  |
| --- |
| **Note**:  This property is overridden to "**true**" automatically by the [NinjaScript Code Wizard](https://ninjatrader.com/es/support/helpGuides/nt8/ns_wizard.htm).  You will need to remove the property to return to the default value or manually set it to false to disable this behavior |

|  |
| --- |
| **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**

IsSuspendedWhileInactive

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         IsSuspendedWhileInactive = true;     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **IsSuspendedWhileInactive** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ischartonly.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/paintpricemarkers.htm) |

**Definition**

Prevents OnBarUpdate from being raised while the indicators display is not in use.  Enabling this property in your indicator helps save CPU cycles while the indicator is suspended and not in use by a user.  Once the indicator is in a state that would no longer be considered suspended, the historical OnBarUpdate() events will be triggered allowing the indicator to catch up to current real-time values.

Suspension occurs in the following scenarios:

•Minimized Chart

•Minimized Market Analyzer

•Minimized Hot List Analyzer

•Minimized SuperDOM

•Background tabs of above features are considered "minimized"

•Inactive workspaces in the background

|  |
| --- |
| **Note**:  Since events in OnBarUpdate() will not be processed while the indicator is suspended, internal NinjaScript functions such as [Alert()](https://ninjatrader.com/es/support/helpGuides/nt8/alert.htm), [PlaySound()](https://ninjatrader.com/es/support/helpGuides/nt8/playsound.htm), [Share()](https://ninjatrader.com/es/support/helpGuides/nt8/share.htm), [Print()](https://ninjatrader.com/es/support/helpGuides/nt8/print.htm), etc - or any other method that would be used to notify a user of activity will **NOT** be processed until the indicator is un-suspended. |

**Scenarios where suspension will not occur**

The **IsSuspendedWhileInactive** property will be ignored and real-time events will be processed as normal under the following cases:

•Indicators running in [Automated NinjaScript Strategies](https://ninjatrader.com/es/support/helpGuides/nt8/running_a_ninjascript_strategy.htm)

•Indicators which have [manually configured alerts](https://ninjatrader.com/es/support/helpGuides/nt8/alerts_dialog.htm)

•Indicators which have been [manually attached to orders](https://ninjatrader.com/es/support/helpGuides/nt8/attachingorderstoindicators.htm)

**Property Value**

This property returns **true** if indicator can take advantage of suspension optimization; otherwise, **false**. Default set to **false**.

|  |
| --- |
| **Note**:  This property is overridden to "**true**" automatically by the [NinjaScript Code Wizard](https://ninjatrader.com/es/support/helpGuides/nt8/ns_wizard.htm).  You will need to remove the property to return to the default value or manually set it to false to disable this behavior |

|  |
| --- |
| **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**

IsSuspendedWhileInactive

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         IsSuspendedWhileInactive = true;     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **PaintPriceMarkers** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/issuspendedwhileinactive.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/showtransparentplotsindatabox.htm) |

**Definition**

If true, any indicator plot values display price markers in the y-axis.

**Property Value**

This property returns **true** if the indicator plot values display in the y-axis; 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**

PaintPriceMarkers

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         PaintPriceMarkers = true; // Indicator plots values display in the y-axis             AddPlot(Brushes.Orange, "SMA");     } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **ShowTransparentPlotsInDataBox** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/paintpricemarkers.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/market_analyzer_column.htm) |

**Definition**

Determines if plot(s) values which are set to a Transparent brush display in the chart data box.  The default behavior is to hide any plots which have been configured as a Transparent brush, however this behavior can be changed by setting **ShowTransparentPlotsInDataBox** to **true** on the indicator.

**Property Value**

This property returns **true** if transparent indicator plot(s) values display in the chart data box; otherwise, **false**. Default set to **false**.

|  |
| --- |
| **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**

ShowTransparentPlotsInDataBox

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         ShowTransparentPlotsInDataBox = true;           AddPlot(Brushes.Transparent, "MyPlot");     } } | |
| **Navigation:**  »No topics above this level«  **Set Up** | | [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout.htm) |

The first step in creating a custom indicator is to use the custom indicator wizard. The wizard will generate the required NinjaScript code that will serve as the foundation for your custom indicator.

1. Within the NinjaTrader Control Center window select the Tools-->New NinjaScript-->Indicator... menu

2. Press the "Next >" button

**Defining Indicator Properties and Name**  
Below you will define your indicators name and several indicator properties.

//screen shot

3. Enter the information as shown above

4. We have checked the option "Overlay on price" which means that this will be a price overlay indicator (will plot on top of the price data instead of in its own indicator panel)

5. Press the "Next >" button

**Defining Input Parameters**

Below you will define your indicator's input parameters. These are any parameters that can be changed by the user and used in the calculation of the indicators value(s). If your indicator does not require any parameters leave the "Name" fields blank.

//screen shot

6. Enter the information as shown above  
7. Press the "Next >" button

**Defining Plots**

Below you will define how your indicator is plotted on a chart.

//screen shot

8. Enter the information as shown above.

9. Press the "Next >" button

**Defining Oscillator Lines**

Below you will define any required oscillator lines. This would be the "Zero" line in the CCI indicator for example. If your indicator does not require any oscillator lines leave the "Name" fields blank.

//screen shot

10. Enter the information as shown above (make sure the "Name" fields are blank since we do not need an oscillator line for a simple moving average indicator)

11. Press the "Next >" button

12. We are now finished entering in our indicator set up information. Press "Finish" button.

\* At any time, you can press the "Generate" button in the wizard if you do not need to go through each of the wizard steps.

You will now see the NinjaScript Editor preloaded with NinjaScript code generated by the wizard. It should look something like the image below.

//screen shot

Your primary area of concern will be sections "2" and "3".

1. This section provides the indicator with the chart display name and description used in the Indicator Dialog window.

2. The OnStateChange() section is processed only once when the indicator is initially loaded (added to a chart for example) and can be used to set up any indicator configuration requirements. You can see that this wizard generated code that added the plot color and style and set the "Overlay" property which we configured in the wizard in step 4 above. The default value of the "Period" parameter is set in the "Variables" region of the code above OnStateChange().

3. This is the section that is called on each bar update (incoming tick) and is where you will enter your indicator logic

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) >  **AddPlot()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/lines.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/areplotsconfigurable.htm) |

**Definition**

Adds plot objects that define how an indicator or strategy data series render on a chart. When this method is called to add a plot, an associated [Series<double>](https://ninjatrader.com/es/support/helpGuides/nt8/seriest.htm) object is created held in the [Values](https://ninjatrader.com/es/support/helpGuides/nt8/value.htm) collection.

|  |
| --- |
| **Note:**  Plots are **ONLY** visible from the UI property grid when AddPlot() is called from **State.SetDefaults**. If your indicator or strategy dynamically adds plots during **State.Configure**, you will **NOT** have an opportunity to select the plot or to set the plot configuration via the UI.  Alternatively, you may use custom public [Brush](https://ninjatrader.com/es/support/helpGuides/nt8/brushes.htm), [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm), or **PlotStyle** properties which are accessible in **State.SetDefaults** and pass those values to AddPlot() during**State.Configure**.  Calling AddPlot() in this manner should be reserved for special cases.  Please see the examples below. |

**Methods and Properties**

|  |  |
| --- | --- |
| [ArePlotsConfigurable](https://ninjatrader.com/es/support/helpGuides/nt8/areplotsconfigurable.htm) | Determines if the plot(s) used in an indicator are configurable within the indicator dialog window. |
| [Displacement](https://ninjatrader.com/es/support/helpGuides/nt8/displacement.htm) | An offset value that shifts the visually displayed value of an indicator. |
| [PlotBrushes](https://ninjatrader.com/es/support/helpGuides/nt8/plotbrushes.htm) | Holds an array of color series objects holding historical bar colors. |
| [Plots](https://ninjatrader.com/es/support/helpGuides/nt8/plots.htm) | A collection holding all of the Plot objects that define their visualization characteristics. |

**Syntax**

AddPlot(Brush brush, string name)  
AddPlot(Stroke stroke, PlotStyle plotStyle, string name)

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

**Parameters**

|  |  |
| --- | --- |
| brush | A Brush object used to construct the plot |
| name | A string representing the name of the plot |
| plotStyle | A PlotStyle object used to construct the style of the plot    Possible values:   PlotStyle.Bar PlotStyle.Block PlotStyle.Cross PlotStyle.Dot PlotStyle.Hash PlotStyle.HLine PlotStyle.Line  PlotStyle.PriceBox PlotStyle.Square PlotStyle.TriangleDown PlotStyle.TriangleLeft PlotStyle.TriangleRight PlotStyle.TriangleUp |
| stroke | A Stroke object used to construct the plot |

|  |
| --- |
| **Tips:**  1.We suggest using the NinjaScript wizard to generate your plots.  2.[Plot](https://ninjatrader.com/es/support/helpGuides/nt8/plots.htm) objects **DO NOT** hold the actual script values. They simply define how the script's values are plotted on a chart.  3.A script may calculate multiple values and therefore hold multiple plots to determine the display of each calculated value. Script values are held in the script's [Values](https://ninjatrader.com/es/support/helpGuides/nt8/value.htm) collection.  4.If you script calls AddPlot() multiple times, then multiple Values series are added per the "three value series" example below  5.For [MultiSeries scripts](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm), plots are synched to the primary series of the NinjaScript object. |

**Examples**

| ns | **Indicator using various AddPlot() signatures** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "Examples Indicator";       // Adds a blue line style plot     AddPlot(Brushes.Blue, "myPlot");       // Adds a blue historgram style plot     AddPlot(new Stroke(Brushes.Blue), PlotStyle.Bar, "myPlot");   } } |

| ns | **Indicator which adds three value series** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "Examples Indicator";       // Add three plots and associated Series<double> objects     AddPlot(Brushes.Blue, "PlotA");     // Defines the plot for Values[0]     AddPlot(Brushes.Red, "PlotB");     // Defines the plot for Values[1]     AddPlot(Brushes.Green, "PlotC");   // Defines the plot for Values[2]   } } protected override void OnBarUpdate() {   Values[0][0] = Median[0];   // Blue "Plot A"   Values[1][0] = Low[0];       // Red "Plot B"   Values[2][0] = High[0];     // Green "Plot C" } |

| ns | **Indicator which dynamically adds a plot in State.Configure** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name                 = "Examples Indicator";       // logical property which user can set     UseSpecialMode   = false;     // Default brush selection pushed to the UI     MyBrush = Brushes.Red;   }   else if (State == State.Configure)   {     // if user enables logical property     if (UseSpecialMode)     {         // add plot using default selected brush and special plot name         AddPlot(MyBrush, "My Special Plot");     }     else     {         // otherwise use default selected brush and regular plot name         AddPlot(MyBrush, "My Regular Plot");     }   } }   protected override void OnBarUpdate() {   if (UseSpecialMode)     Value[0] = Close[0] + High[0] / 2;     else Value[0] = Close[0] \* TickSize / 2; }   [XmlIgnore] public Brush MyBrush { get; set; }   public bool UseSpecialMode { get; set; } |