|  |  |
| --- | --- |
| **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.Line()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/horizontalline.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/line.htm) |

**Definition**

Draws a line between two points.

**Method Return Value**

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

**Syntax**

Draw.Line(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush)  
Draw.Line(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, DashStyleHelper dashStyle, int width)  
Draw.Line(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, DashStyleHelper dashStyle, int width)  
Draw.Line(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.Line(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.Line(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, string templateName)  
Draw.Line(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, string templateName)  
Draw.Line(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, bool isGlobal, string templateName)  
Draw.Line(NinjaScriptBase owner, string tag, bool isAutoScale, 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)) |
| 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 line from 10 bars back to the current bar // with a width of 2 pixels Draw.Line(this, "tag1", false, 10, 1000, 0, 1001, Brushes.LimeGreen, DashStyleHelper.Dot, 2); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Educational Resources](https://ninjatrader.com/es/support/helpGuides/nt8/educational_resources.htm) >  **Working with Brushes** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/educational_resources.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_chart_object_coordinates.htm) |

In order to achieve custom rendering for various chart related objects, a Brush is used to "paint" an area or another chart object.  There are a number of different brushes which are available through the .NET Framework, where the most common type of brush is a [SolidColorBrush](https://msdn.microsoft.com/en-us/library/system.windows.media.solidcolorbrush(v=vs.110).aspx) which is used to paint an area with a single solid color.

|  |
| --- |
| **Notes**:  The following document is written in sequential fashion, starting with the most simple concepts, to the more advance topics.  The majority of the brushes discussed in this document will be referred to as "**WPF" brushes** which exist in the System.Windows.Media namespace, however there are also **"SharpDX" brushes** which exist in the 3rd party SharpDX.Direct2D1 nampspace used for advanced chart rendering.  Advanced brush types should **ONLY** be used by experienced programmers familiar with .NET graphics functionality. |

tog_minus        [Understanding predefined brushes](javascript:HMToggle('toggle','Understandingpredefinedbrushes','Understandingpredefinedbrushes_ICON'))

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Using Predefined Brushes**  For convenience, the .NET Framework supplies a collection of static predefined Brushes, such as Red or Green.  The advantage to using these brushes is that they are readily available, properly named to quickly find a simple color value, and can be reused on-the-fly without having to recreate an instance of the brush at run time, and do not need to be otherwise managed.  There are 256 predefined named brushes which are available in the Brushes class.  You can browse this list in the NinjaScript editor just by typing Brushes. and using Intelliprompt to find the desired named brush of your choice.     |  | | --- | | **Note**:   Since predefined brushes are static, properties of the brush object (such as Color, Opacity, etc.) **CANNOT** be modified.  However, this also means predefined brushes are thread-safe and do **NOT** need to be frozen.  For customizing and freezing a brush, please see the section below on *Creating a Custom Solid Color Brush*. |       Brushes       |  | | --- | | **Tip**:  You can also find a list of these predefined brushes as well as their hexadecimal value on the MSDN article for the [Brushes Class](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes(v=vs.110).aspx) |        | ns |  | | --- | --- | | // set the chart's background color to a predefined "Blue" brush BackBrush = Brushes.Blue;   //draw a line using a predefined "LimeGreen" brush. Draw.Line(this, "tag1", false, 10, 1000, 0, 1001, Brushes.LimeGreen, DashStyleHelper.Dot, 2); | | |

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

tog_minus        [Understanding custom brushes](javascript:HMToggle('toggle','Understandingcustombrushes','Understandingcustombrushes_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Creating a Custom Solid Color Brush**  In cases where you would like more specific color than one of the predefined brushes, you can optionally create your own **Brush** object to be used for custom rendering.  In order to achieve this, you will need to initiate your own custom brush object, where you can then specify your color using RGB (red, green, blue) values [Color.FromRgb()](https://msdn.microsoft.com/en-us/library/system.windows.media.color.fromrgb(v=vs.110).aspx).     |  | | --- | | **Notes**:  •Anytime you create a custom brush that will be used by NinjaTrader rendering it must be frozen using the .[Freeze()](https://msdn.microsoft.com/en-us/library/ms557735(v=vs.110).aspx)  method due to the multi-threaded nature of NinjaTrader.  •You may have up to 65535 unique Brush instances, therefore, using static predefined brushes (as in the section above) 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. |        | ns |  | | --- | --- | | // initiate new solid color brush with custom blue color Brush myBrush = new SolidColorBrush(Color.FromRgb(56, 120, 153)); myBrush.Freeze();   Draw.Line(this, "tag1", true, 10, 1000, 0, 1001, myBrush, DashStyleHelper.Dot, 2); | |      |  | | --- | | **Warning**:  If you do not call .[Freeze()](https://msdn.microsoft.com/en-us/library/ms557735(v=vs.110).aspx) on a custom defined brush **WILL**eventually result in threading errors should you try to modify or access that brush after it is defined. |       **Creating a Transparent Solid Color Brush**  You can create a transparent brush using the [Color.FromArgb()](https://msdn.microsoft.com/en-us/library/system.windows.media.color.fromargb(v=vs.110).aspx) where the A parameter defines alpha transparency.     |  | | --- | | **Note**:   Anytime you create a custom brush that will be used by NinjaTrader rendering it must be frozen using the .[Freeze()](https://msdn.microsoft.com/en-us/library/ms557735(v=vs.110).aspx)  method due to the multi-threaded nature of NinjaTrader. |        | ns |  | | --- | --- | | // initiate new solid color brush which has an alpha (transparency) value of 100 MyBrush = new SolidColorBrush(Color.FromArgb(100, 56, 120, 153)); myBrush.Freeze();   Draw.Line(this, "tag1", true, 10, 1000, 0, 1001, myBrush, DashStyleHelper.Dot, 2); | |        |  | | --- | | **Warning**:  If you do not call .[Freeze()](https://msdn.microsoft.com/en-us/library/ms557735(v=vs.110).aspx) on a custom defined brush **WILL**eventually result in threading errors should you try to modify or access that brush after it is defined. | |

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

tog_minus        [Using brushes defined on the user interface](javascript:HMToggle('toggle','Userdefinedbrushes','Userdefinedbrushes_ICON'))

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Saving a Brush as a user defined property (Serialization)**  If you would like a brush to become a public UI property, meaning the brush can be set up and defined by a user during configuration, it is important to be able to save the user's brush selection in order to restore that brush either from a workspace or from a template file at a later time.  Saving a custom defined user input is done through a concept of [Serialization](https://msdn.microsoft.com/en-us/library/ms233843.aspx) which writes the object and its value to a .xml file.  This process normally works fine for a simple user defined value type (such as a double or an int) but for more complex types such as Brushes, the object itself cannot be serialized directly to the .xml file and will result in errors upon saving the indicator or strategy to a workspace or template file.  The example below will demonstrate and explain how to properly store a user define brush input which will be correctly serialized.    In order to achieve the desired behavior of saving the user defined brush input, we will add the [XmlIgnore](https://msdn.microsoft.com/en-us/library/system.xml.serialization.xmlignoreattribute(v=vs.110).aspx) property attribute to the public brush resource, which essentially tells the serialization routine to ignore this property.     | ns | | --- | | [XmlIgnore] public Brush MyBrush { get; set; } |       In its place, we create a new public string called "MyBrushSerialize" which will convert the public "MyBrush" to a string type which can then be processed by the serialization routines.  We also add the [Browsable(false)](https://msdn.microsoft.com/en-us/library/system.componentmodel.browsableattribute(v=vs.110).aspx) attribute to this public string to prevent this property from showing up on the UI, which is of no value to the end user.     | ns | | --- | | [Browsable(false)] public string MyBrushSerialize {   get { return Serialize.BrushToString(MyBrush); }   set { MyBrush = Serialize.StringToBrush(value); } } |        |  | | --- | | **Tip**: For a complete example of **User Definable Color Inputs**, please see the reference sample [here](https://ninjatrader.com/es/support/helpGuides/nt8/user_definable_color_inputs.htm). |       **Adding a User Defined Brush to the Color Picker**  You can optionally define a custom brush to be added to the standard color picker by using a [CustomBrush] attribute to a public brush.  The CustomBrush attribute will then add it to the color picker menu for that indicator when you look through the plots, lines, or other brushes from the indicators configured menu and will be listed toward the top of the list (as pictured below)     | ns | | --- | | [CustomBrush] public Brush MyBrush {   get { return new SolidColorBrush(Color.FromRgb(25, 175, 185)); }   set { } } |     custom_brush |

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

tog_minus        [Using advanced brush types (SharpDX)](javascript:HMToggle('toggle','AdvancedBrushTypesSharpDX','AdvancedBrushTypesSharpDX_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Understanding SharpDX Brushes**  While the majority of the NinjaTrader platform's UI is **WPF**, under the hood, chart's use a **DirectX API** for faster performance.  To render custom objects to a chart during [OnRender()](https://ninjatrader.com/es/support/helpGuides/nt8/onrender.htm), a particular **SharpDX** **Brush** object must be implemented which reside in the **SharpDX.Direct2D1** namespace.   These brushes can then be passed as arguments to the **SharpDX** [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/rendertarget.htm) methods such [FillRectangle()](https://ninjatrader.com/es/support/helpGuides/nt8/fillrectangle.htm), [DrawLine()](https://ninjatrader.com/es/support/helpGuides/nt8/drawline2.htm), etc.  While **SharpDX Brushes** behave much the same as previously discussed **WPF** **Brushes**, there are a few special considerations you must take as detailed in the following sections.     |  | | --- | | **Note**:  The **SharpDX Brushes** used in [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/rendertarget.htm) methods should **NOT** be confused with the **WPF Brushes** used with [DrawingTool Draw](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) methods. |       **Creating a SharpDX Brush**  A [SharpDX Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) must be created either in **OnRender()** or **RenderTargetChanged()**.  If you have custom brushes which may be changed on various conditions such as in OnBarUpdate() or by a user during OnStateChange(), or you are pre-computing a custom brush for performance optimization, you will need to ensure the actual SharpDX instance is updated in OnRender() or RenderTargetChange().     |  | | --- | | **Warning**:  Each DirectX render target requires its own brushes. You **MUST** create brushes directly in **OnRender()** or using **OnRenderTargetChanged()**.  If you do not you will receive an error at runtime similar to:   ***"A direct X error has occured while rendering the chart: HRESULT: [0x88990015], Module: [SharpDX.Direct2D1], ApiCode: [D2DERR\_WRONG\_RESOURCE\_DOMAIN/WrongResourceDomain], Message: The resource was realized on the wrong render target. : Each DirectX render target requires its own brushes. You must create brushes directly in OnRender() or using OnRenderTargetChanged().***    Please see [OnRenderTargetChanged()](https://ninjatrader.com/es/support/helpGuides/nt8/onrendertargetchanged.htm) for examples of a brush that needs to be recalculated, or [OnRender()](https://ninjatrader.com/es/support/helpGuides/nt8/onrender.htm) for an example of recreating a static brush. |        | ns | | --- | | // use predefined "Blue" SharpDX Color SharpDX.Direct2D1.SolidColorBrush solidBlueDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.Blue);   // create custom Brush using a "Red" SharpDX Color with "Alpha" (0.100f) transparency/opacity SharpDX.Direct2D1.SolidColorBrush transparentRedDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, new SharpDX.Color4(new SharpDX.Color3(220f, 0f, 0f), 0.100f)); |       **Converting to SharpDX Brush**  For convenience, you can convert a computed WPF Brush to a [SharpDX Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) using the [ToDxBrush(](https://ninjatrader.com/es/support/helpGuides/nt8/dxextensions_todxbrush.htm)) extension method.     |  | | --- | | **Warning**:  Converting **ToDxBrush()** can result in performance issues depending on the number of brushes being used. If you experience performance issues with your custom **SharpDX** rendering, you should favor using **SharpDX** brushes directly instead of converting the brush using **ToDxBrush().** |        | ns | | --- | | // convert predefined WPF "Blue" to SharpDX Brush SharpDX.Direct2D1.Brush blueDXBrush = Brushes.Blue.ToDxBrush(RenderTarget);   // convert the computed WPF Brush to SharpDX Brush SharpDX.Direct2D1.Brush customDXBrush = customWPFBrush.ToDxBrush(RenderTarget); |       **Disposing DXBrush**  Since **SharpDX Brushes** reference unmanaged resources, these brushes should always be [disposed](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase.htm) of after they have been used.     |  | | --- | | **Warning**:  Failing to dispose of a [SharpDX Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) and other unmanaged resources can cause the platform to utilize more memory than necessary. |        | ns | | --- | | customDXBrush.Dipose(); |     **Using Complex Brushes**  In addition to the [SolidColorBrush](https://msdn.microsoft.com/en-us/library/system.windows.media.solidcolorbrush(v=vs.110).aspx) object demonstrated on this page, the .NET Framework provides more complex brushes which have more attributes than just filling an area with a solid color.  Information on these special types of brushes can be found on the MSDN website: [LinearGradientBrush](https://msdn.microsoft.com/en-us/library/system.windows.media.lineargradientbrush(v=vs.110).aspx), [RadialGradientBrush](https://msdn.microsoft.com/en-us/library/system.windows.media.radialgradientbrush(v=vs.110).aspx), [ImageBrush](https://msdn.microsoft.com/en-us/library/system.windows.media.imagebrush(v=vs.110).aspx).    These complex types also have an equivalent found in the**SharpDX SDK Reference**: [SharpDX.Direct2D1.LinearGradientBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_lineargradientbrush.htm), [SharpDX.Direct2D1.RadialGradientBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_radialgradientbrush.htm) |

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

|  |  |
| --- | --- |
| **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)) |
| 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.Polygon()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/pathtool.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/polygon.htm) |

**Definition**

Draws a polygon which can have a user defined set of anchors.

**Method Return Value**

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

**Syntax**

Draw.Polygon(NinjaScriptBase owner, string tag, bool isAutoScale, List<ChartAnchor> chartAnchors, bool isGlobal, string templateName)

Draw.Polygon(NinjaScriptBase owner, string tag, bool isAutoScale, List<ChartAnchor> chartAnchors, Brush brush, DashStyleHelper dashStyle, Brush areaBrush, int areaOpacity)

Draw.Polygon(NinjaScriptBase owner, string tag, bool isAutoScale, int anchor1BarsAgo, double anchor1Y, int anchor2BarsAgo, double anchor2Y, int anchor3BarsAgo, double anchor3Y, int anchor4BarsAgo, double anchor4Y)

Draw.Polygon(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime Anchor1Time, double anchor1Y, DateTime Anchor2Time, double anchor2Y, DateTime Anchor3Time, double anchor3Y, DateTime Anchor4Time, double anchor4Y)

Draw.Polygon(NinjaScriptBase owner, string tag, bool isAutoScale, int anchor1BarsAgo, double anchor1Y, int anchor2BarsAgo, double anchor2Y, int anchor3BarsAgo, double anchor3Y, int anchor4BarsAgo, double anchor4Y, int anchor5BarsAgo, double anchor5Y)

Draw.Polygon(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime Anchor1Time, double anchor1Y, DateTime Anchor2Time, double anchor2Y, DateTime Anchor3Time, double anchor3Y, DateTime Anchor4Time, double anchor4Y, DateTime Anchor5Time, double anchor5Y)

Draw.Polygon(NinjaScriptBase owner, string tag, bool isAutoScale, int anchor1BarsAgo, double anchor1Y, int anchor2BarsAgo, double anchor2Y, int anchor3BarsAgo, double anchor3Y, int anchor4BarsAgo, double anchor4Y, int anchor5BarsAgo, double anchor5Y, int anchor6BarsAgo, double anchor6Y)

Draw.Polygon(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime Anchor1Time, double anchor1Y, DateTime Anchor2Time, double anchor2Y, DateTime Anchor3Time, double anchor3Y, DateTime Anchor4Time, double anchor4Y, DateTime Anchor5Time, double anchor5Y, DateTime Anchor6Time, double anchor6Y)

**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. |
| chartAnchors | A list of the chart anchors |
| anchor1BarsAgo | The bar the first anchor of the object will be drawn at. A value of 10 would be 10 bars ago. |
| anchor2BarsAgo | The bar the second anchor of the object will be drawn at. A value of 10 would be 10 bars ago. |
| anchor3BarsAgo | The bar the third anchor of the object will be drawn at. A value of 10 would be 10 bars ago. |
| anchor4BarsAgo | The bar the forth anchor of the object will be drawn at. A value of 10 would be 10 bars ago. |
| anchor5BarsAgo | The bar the fifth anchor of the object will be drawn at. A value of 10 would be 10 bars ago. |
| anchor6BarsAgo | The bar the sixth anchor of the object will be drawn at. A value of 10 would be 10 bars ago. |
| anchor1Y | The first anchor y value |
| anchor2Y | The second anchor y value |
| anchor3Y | The third anchor y value |
| anchor4Y | The forth anchor y value |
| anchor5Y | The fifth anchor y value |
| anchor6Y | The sixth anchor y value |
| Anchor1Time | The time the first anchor of the object will be drawn at |
| Anchor2Time | The time the second anchor of the object will be drawn at |
| Anchor3Time | The time the third anchor of the object will be drawn at |
| Anchor4Time | The time the forth anchor of the object will be drawn at |
| Anchor5Time | The time the fifth anchor of the object will be drawn at |
| Anchor6Time | The time the sixth anchor of the object will be drawn at |
| areaBrush | The brush used to color draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx)) |
| areaOpacity | Sets the level of transparency for the fill color. Valid values between 0 - 100. (0 = completely transparent, 100 = no opacity) |
| 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 Polygon object based on bars ago and y anchors Draw.Polygon(this, "tag1", false, 20, 194, 10, 184, 13, 176, 25, 182);    // Draws a Polygon object based on a list of anchors with specified times  List<ChartAnchor> anchors = new List<ChartAnchor>();  anchors.Add(new ChartAnchor(new DateTime(2018, 5, 25), 194, ChartControl));  anchors.Add(new ChartAnchor(new DateTime(2018, 6, 12), 184, ChartControl));  anchors.Add(new ChartAnchor(new DateTime(2018, 6, 7), 176, ChartControl));  anchors.Add(new ChartAnchor(new DateTime(2018, 5, 21), 182, ChartControl));    Draw.Polygon(this, "tag1", false, anchors, Brushes.CornflowerBlue, DashStyleHelper.Solid, Brushes.CornflowerBlue, 40); | |
| **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.Ray()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/polygon.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/ray.htm) |

**Definition**

Draws a line which has an infinite end point in one direction.

**Method Return Value**

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

**Syntax**

Draw.Ray(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush)  
Draw.Ray(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, DashStyleHelper dashStyle, int width)  
Draw.Ray(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush)  
Draw.Ray(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, DashStyleHelper dashStyle, int width)  
Draw.Ray(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.Ray(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, DashStyleHelper dashStyle, int width, bool drawOnPricePanel)  
Draw.Ray(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, bool isGlobal, string templateName)  
Draw.Ray(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 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 |
| 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)) |
| 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 lime green ray from 10 bars back through the current bar Draw.Ray(this, "tag1", 10, 1000, 0, 1001, 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) >  **Draw.Rectangle()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ray.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/rectangle.htm) |

**Definition**

Draws a rectangle.

**Method Return Value**

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

**Syntax**

Draw.Rectangle(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush)  
Draw.Rectangle(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush)  
Draw.Rectangle(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, Brush areaBrush, int areaOpacity)  
Draw.Rectangle(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, Brush areaBrush, int areaOpacity)  
Draw.Rectangle(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, bool drawOnPricePanel)  
Draw.Rectangle(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, Brush brush, Brush areaBrush, int areaOpacity, bool drawOnPricePanel)  
Draw.Rectangle(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, Brush brush, Brush areaBrush, int areaOpacity, bool drawOnPricePanel)  
Draw.Rectangle(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int endBarsAgo, double endY, bool isGlobal, string templateName)  
Draw.Rectangle(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)) |
| 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)) |
| 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 |
| 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 blue rectangle from the low 10 bars back to the high of 5 bars back Draw.Rectangle(this, "tag1", 10, Low[10] - TickSize, 5, High[5] + TickSize, Brushes.Blue);   // Draws a blue rectangle from the low 10 bars back to the high of 5 bars back with // a fill color or pale green with a transparency level of 2 Draw.Rectangle(this, "tag1", false, 10, Low[10] - TickSize, 5, High[5] + TickSize, Brushes.PaleGreen, Brushes.PaleGreen, 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.Region()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/rectangle.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/region.htm) |

**Definition**

Draws a region on a chart.

**Method Return Value**

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

**Syntax**

Draw.Region(NinjaScriptBase owner, string tag, int startBarsAgo,  
        int endBarsAgo, ISeries<double> series, double price, Brush areaBrush, int areaOpacity, int displacement = 0)  
Draw.Region(NinjaScriptBase owner, string tag, int startBarsAgo,  
        int endBarsAgo, ISeries<double> series1, ISeries<double> series2, Brush outlineBrush,  
        Brush areaBrush, int areaOpacity, [int displacement])  
Draw.Region(NinjaScriptBase owner, string tag, DateTime startTime,  
        DateTime endTime, ISeries<double> series, double price, Brush areaBrush, int areaOpacity)  
Draw.Region(NinjaScriptBase owner, string tag, DateTime startTime,  
        DateTime endTime, ISeries<double> series1, ISeries<double> series2, 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. |
| 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. |
| endBarsAgo | The end bar (x axis co-ordinate) where the draw object will terminate |
| endTime | The end time where the draw object will terminate |
| series, series1, series2 | Any Series<double> type object such as an indicator, Close, High, Low etc.. The value of the object will represent a y value. |
| price | Any double value |
| outlineBrush | The brush used to color the region outline of draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx)) |
| areaBrush | The brush used to color the fill region area of the draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx)) |
| areaOpacity | Sets the level of transparency for the fill color. Valid values between 0 - 100. (0 = completely transparent, 100 = no opacity) |
| displacement | An optional parameter which will offset the barsAgo value for the Series<double> value used to match the desired [Displacement](https://ninjatrader.com/es/support/helpGuides/nt8/displacement.htm).  Default value is 0. |

**Example**

| ns |
| --- |
| // Draw a region between upper and lower Bollinger bands  Draw.Region(this, "tag1", CurrentBar, 0, Bollinger(2, 14).Upper, Bollinger(2, 14).Lower, null, Brushes.Blue, 50); |

|  |  |
| --- | --- |
| **Tips**: 1. Pass in null to the "outlineColor" parameter if you do not want to have an outline color. 2. If you wanted to fill a region between a value (20 period simple moving average) and the upper edge of the chart, pass in an extreme value to the "y" parameter such as 1000000.  3. Should you be drawing regions based on Series<double> objects instead of indicator plots, be sure to create the Series<double> with the MaximumBarsLookBack.Infinite parameter if the region you are drawing would be maintained on the chart for more than 256 bars back. | |
| **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.RegionHighlightX()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/region.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/regionhighlightx.htm) |

**Definition**

Draws a region highlight x on a chart.

**Method Return Value**

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

**Syntax**

Draw.RegionHighlightX(NinjaScriptBase owner, string tag, DateTime startTime, DateTime endTime, Brush brush)  
Draw.RegionHighlightX(NinjaScriptBase owner, string tag, int startBarsAgo, int endBarsAgo, Brush brush)  
Draw.RegionHighlightX(NinjaScriptBase owner, string tag, DateTime startTime, DateTime endTime, Brush brush, Brush areaBrush, int areaOpacity)  
Draw.RegionHighlightX(NinjaScriptBase owner, string tag, int startBarsAgo, int endBarsAgo, Brush brush, Brush areaBrush, int areaOpacity)  
Draw.RegionHighlightX(NinjaScriptBase owner, string tag, DateTime startTime, DateTime endTime, bool isGlobal, string templateName)  
Draw.RegionHighlightX(NinjaScriptBase owner, string tag, int startBarsAgo, int endBarsAgo, 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. |
| 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. |
| endBarsAgo | The end bar (x axis co-ordinate) where the draw object will terminate |
| endTime | The end time 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)) |
| 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)) |
| 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 | |
| --- | --- |
| // Fills in the region between the startBar and endBar Draw.RegionHighlightX(this, "tag1", 10, 0, 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.RegionHighlightY()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/regionhighlightx.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/regionhighlighty.htm) |

**Definition**

Draws a region highlight y on a chart.

**Method Return Value**

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

**Syntax**

Draw.RegionHighlightY(NinjaScriptBase owner, string tag, double startY, double endY, Brush brush)  
Draw.RegionHighlightY(NinjaScriptBase owner, string tag, bool isAutoScale, double startY, double endY, Brush brush, Brush areaBrush, int areaOpacity)  
Draw.RegionHighlightY(NinjaScriptBase owner, string tag, double startY, 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. |
| startY | The starting y value co-ordinate where the draw object will be drawn |
| endY | The ending y value co-ordinate where the draw object will be drawn |
| 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)) |
| 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)) |
| 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 | |
| --- | --- |
| // Fills in the region between the startY and endY Draw.RegionHighlightY(this, "tag1", true, High[0], Low[0], Brushes.Blue, Brushes.Green, 20); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.RegressionChannel()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/regionhighlighty.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/regressionchannel.htm) |

**Definition**

Draws a regression channel.

**Method Return Value**

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

**Syntax**  
Draw.RegressionChannel(NinjaScriptBase owner, string tag, int startBarsAgo, int endBarsAgo, Brush brush)  
Draw.RegressionChannel(NinjaScriptBase owner, string tag, DateTime startTime, DateTime endTime, Brush brush)  
Draw.RegressionChannel(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, int endBarsAgo, Brush upperBrush, DashStyleHelper upperDashStyleHelper, int upperWidth, Brush middleBrush, DashStyleHelper middleDashStyleHelper, int middleWidth, Brush lowerBrush, DashStyleHelper lowerDashStyleHelper, int lowerWidth)  
Draw.RegressionChannel(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, DateTime endTime, Brush upperBrush, DashStyleHelper upperDashStyleHelper, int upperWidth, Brush middleBrush, DashStyleHelper middleDashStyleHelper, int middleWidth, Brush lowerBrush, DashStyleHelper lowerDashStyleHelper, int lowerWidth)  
Draw.RegressionChannel(NinjaScriptBase owner, string tag, int startBarsAgo, int endBarsAgo, bool isGlobal, string templateName)  
Draw.RegressionChannel(NinjaScriptBase owner, string tag, DateTime startTime, DateTime endTime, 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. |
| endBarsAgo | The end bar (x axis co-ordinate) where the draw object will terminate |
| endTime | The end time 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)) |
| upperDashStyle, middleDashStyle, lowerDashStyle | DashStyleHelper.Dash DashStyleHelper.DashDot DashStyleHelper.DashDotDot DashStyleHelper.Dot DashStyleHelper.Solid    **Note**: Fancier DashStyles like DashDotDot will require more resources than simple DashStyles like Solid. |
| upperBrush, middleBrush, lowerBrush | The line colors ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx)) |
| upperWidth, middleWidth,  lowerWidth | The line width |
| 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 regression channel from the low 10 bars back to the high of 5 bars back Draw.RegressionChannel(this, "tag1", 10, 0, 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.RiskReward()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/regressionchannel.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/riskreward.htm) |

**Definition**

Draws a risk/reward on a chart.

**Method Return Value**

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

**Syntax**

Draw.RiskReward(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime entryTime, double entryY, DateTime endTime, double endY, double ratio, bool isStop)  
Draw.RiskReward(NinjaScriptBase owner, string tag, bool isAutoScale, int entryBarsAgo , double entryY, int endBarsAgo, double endY, double ratio, bool isStop)  
Draw.RiskReward(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime entryTime, double entryY, DateTime endTime, double endY, double ratio, bool isStop, bool isGlobal, string templateName)  
Draw.RiskReward(NinjaScriptBase owner, string tag, bool isAutoScale, int entryBarsAgo , double entryY, int endBarsAgo, double endY, double ratio, bool isStop, 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. |
| entryTime | The time where the draw object's entry will be drawn. |
| entryBarsAgo | The starting bar (x axis co-ordinate) where the draw object's entry will be drawn. For example, a value of 10 would paint the draw object 10 bars back. |
| entryY | The y value co-ordinate where the draw object's entry price 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 starting y value co-ordinate where the draw object will be drawn |
| ratio | An double value determining the calculated ratio between the risk and reward based on the entry point. Example: reward : risk is ratio of 1.0 |
| isStop | A bool value, when true will use the endTime / endBarsAgo and endY to set the stop and will automatically calculate the target based off the ratio value. When false, will set the target and will automatically calculate the stop based off the ratio value. |
| 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 | |
| --- | --- |
| // draw a risk/reward tool starting from the current bar to 10 bars ago // with calcuate a ratio of 2 based on stop level Draw.RiskReward(this, "tag1", false, 0, High[0], 10, Low[0], 2, 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) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.Ruler()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/riskreward.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/ruler.htm) |

**Definition**

Draws a ruler.

**Method Return Value**

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

**Syntax**

Draw.Ruler(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, int textBarsAgo, double textY)  
Draw.Ruler(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, DateTime textTime, double textY)  
Draw.Ruler(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int endBarsAgo, double endY, int textBarsAgo, double textY, bool isGlobal, string templateName)  
Draw.Ruler(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime endTime, double endY, DateTime textTime, double textY, 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 |
| textBarsAgo | The number of bars ago (x value) of the 3rd anchor point |
| textTime | The time of the 3rd anchor point |
| textY | 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) |

**Example**

| ns | |
| --- | --- |
|  | // Draws a ruler measuring the primary bar series Draw.Ruler(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.Square()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ruler.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/square.htm) |

**Definition**

Draws a square.

**Method Return Value**

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

**Syntax**

Draw.Square(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush)  
Draw.Square(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush)  
Draw.Square(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush, bool drawOnPricePanel)  
Draw.Square(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush, bool drawOnPricePanel)  
Draw.Square(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, bool isGlobal, string templateName)  
Draw.Square(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)) |
| 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 square is tied to the chart's BarWidth and thus will scale automatically as the chart is resized |

**Examples**

| ns | |
| --- | --- |
| // Paints a red square on the current bar 1 tick below the low Draw.Square(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.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)) |
| 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)) |
| 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)) |
| 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)) |
| 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.TextFixed()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/text.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/textfixed.htm) |

**Definition**

Draws text in one of 5 available pre-defined fixed locations on panel 1 (price panel) of a chart. Please note the [Z-Order](https://ninjatrader.com/es/support/helpGuides/nt8/zordertype.htm) is internally set for the method to always be drawn on top.

**Method Return Value**

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

**Syntax**

Draw.TextFixed(NinjaScriptBase owner, string tag, string text, TextPosition textPosition, Brush textBrush, SimpleFont font, Brush outlineBrush, Brush areaBrush, int areaOpacity)  
Draw.TextFixed(NinjaScriptBase owner, string tag, string text, TextPosition textPosition)  
Draw.TextFixed(NinjaScriptBase owner, string tag, string text, TextPosition textPosition, 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. |
| text | The text you wish to draw |
| TextPosition | TextPosition.BottomLeft  TextPosition.BottomRight  TextPosition.Center  TextPosition.TopLeft  TextPosition.TopRight |
| textBrush | The brush used to color the text of the draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx)) |
| font | A Simple Font object |
| 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)) |
| 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)) |
| 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 in the upper right corner of panel 1 Draw.TextFixed(this, "tag1", "Text to draw", TextPosition.TopRight); |

|  |
| --- |
| **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.TextFixed(this, "myTextFixed", "Hello world!", TextPosition.BottomRight, ChartControl.Properties.ChartText,   ChartControl.Properties.LabelFont, Brushes.Blue, Brushes.Transparent, 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.TimeCycles()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/textfixed.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/timecycles.htm) |

**Definition**

Draws a time cycle based on two points.

**Method Return Value**

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

**Syntax**

Draw.TimeCycles(NinjaScriptBase owner, string tag, int startBarsAgo, int endBarsAgo, double endY, Brush brush, bool drawOnPricePanel)

Draw.TimeCycles(NinjaScriptBase owner, string tag, int startBarsAgo, int endBarsAgo, bool isGlobal, string templateName)

Draw.TimeCycles(NinjaScriptBase owner, string tag, DateTime startTime, DateTime endTime, Brush brush, bool drawOnPricePanel)

Draw.TimeCycles(NinjaScriptBase owner, string tag, DateTime startTime, DateTime endTime, bool isGlobal, string templateName)

Draw.TimeCycles(NinjaScriptBase owner, string tag, DateTime startTime, DateTime endTime, Brush brush, Brush areaBrush, int areaOpacity)

Draw.TimeCycles(NinjaScriptBase owner, string tag, int startBarsAgo, int endBarsAgo, Brush brush, Brush areaBrush, int areaOpacity)

Draw.TimeCycles(NinjaScriptBase owner, string tag, int startBarsAgo, int endBarsAgo, Brush brush, Brush areaBrush, int areaOpacity, bool drawOnPricePanel)

Draw.TimeCycles(NinjaScriptBase owner, string tag, DateTime startTime, DateTime endTime, Brush brush, Brush areaBrush, int areaOpacity, bool drawOnPricePanel)

Draw.TimeCycles(NinjaScriptBase owner, string tag, DateTime startTime, DateTime endTime, Brush brush)

Draw.TimeCycles(NinjaScriptBase owner, string tag, int startBarsAgo, int endBarsAgo, Brush brush)

**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. |
| 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 |
| endBarsAgo | The end bar (x axis co-ordinate) where the draw object will terminate |
| endTime | The end time 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)) |
| 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 Time Cycles object based on 10 bars back to the current bar that is cornflower blue with an opacity of 40 Draw.TimeCycles(this, "tag1", 0, 10, Brushes.CornflowerBlue, Brushes.CornflowerBlue, 40); | |
| **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.TrendChannel()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/timecycles.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/trendchannel.htm) |

**Definition**

Draws a trend channel.

**Method Return Value**

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

**Syntax**

Draw.TrendChannel(NinjaScriptBase owner, string tag, bool isAutoScale, int anchor1BarsAgo, double anchor1Y, int anchor2BarsAgo, double anchor2Y, int anchor3BarsAgo, double anchor3Y)  
Draw.TrendChannel(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime anchor1Time, double anchor1Y, DateTime anchor2Time, double anchor2Y, DateTime anchor3Time, double anchor3Y)  
Draw.TrendChannel(NinjaScriptBase owner, string tag, bool isAutoScale, int anchor1BarsAgo, double anchor1Y, int anchor2BarsAgo, double anchor2Y, int anchor3BarsAgo, double anchor3Y, bool isGlobal, string templateName)  
Draw.TrendChannel(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 |
| 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 trend channel Draw.TrendChannel(this, "tag1", true, 10, Low[10], 0, High[0], 10, High[10] + 5 \* TickSize); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.Triangle()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/trendchannel.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/triangle.htm) |

**Definition**

Draws a triangle.

**Method Return Value**

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

**Syntax**

Draw.Triangle(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int middleBarsAgo, double middleY, int endBarsAgo, double endY, Brush brush)  
Draw.Triangle(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime middleTime, double middleY, DateTime endTime, double endY, Brush brush)  
Draw.Triangle(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int middleBarsAgo, double middleY, int endBarsAgo, double endY, Brush brush, Brush areaBrush, int areaOpacity)  
Draw.Triangle(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime midTime, double middleY, DateTime endTime, double endY, Brush brush, Brush areaBrush, int areaOpacity)  
Draw.Triangle(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int middleBarsAgo, double middleY, int endBarsAgo, double endY, Brush brush, bool drawOnPricePanel)  
Draw.Triangle(NinjaScriptBase owner, string tag, bool isAutoScale, int startBarsAgo, double startY, int middleBarsAgo, double middleY, int endBarsAgo, double endY, Brush brush, Brush areaBrush, int areaOpacity, bool drawOnPricePanel)  
Draw.Triangle(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime startTime, double startY, DateTime midTime, double middleY, DateTime endTime, double endY, Brush brush, Brush areaBrush, int areaOpacity, bool drawOnPricePanel)  
Draw.Triangle(NinjaScriptBase owner, string tag, int startBarsAgo, double startY, int middleBarsAgo, double middleY, int endBarsAgo, double endY, bool isGlobal, string templateName)  
Draw.Triangle(NinjaScriptBase owner, string tag, DateTime startTime, double startY, DateTime middleTime, double middleY, 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 |
| 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 |
| middleBarsAgo | The number of bars ago (x value) of the 2nd anchor point |
| midTime | The time of the 2nd anchor point |
| middleY | The y value of the 2nd anchor point |
| endBarsAgo | The number of bars ago (x value) of the 3rd anchor point |
| endTime | The time of the 3rd anchor point |
| endY | The y value of the 3rd anchor point |
| 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)) |
| 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)) |
| 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 |
| 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 | |
| --- | --- |
| // Paints a blue triangle on the chart Draw.Triangle(this, "tag1", 4, Low[4], 3, High[3], 1, Low[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) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.TriangleDown()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/triangle.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/triangledown.htm) |

**Definition**

Draws a triangle pointing down.

**Method Return Value**

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

**Syntax**

Draw.TriangleDown(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush)  
Draw.TriangleDown(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush)  
Draw.TriangleDown(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush, bool drawOnPricePanel)  
Draw.TriangleDown(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush, bool drawOnPricePanel)  
Draw.TriangleDown(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, bool isGlobal, string templateName)  
Draw.TriangleDown(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)) |
| 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 triangle is tied to the chart's BarWidth and thus will scale automatically as the chart is resized |

**Examples**

| ns | |
| --- | --- |
| // Paints a red triangle pointing down on the current bar 1 tick below the low Draw.TriangleDown(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.TriangleUp()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/triangledown.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/triangleup.htm) |

**Definition**

Draws a triangle pointing up.

**Method Return Value**

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

**Syntax**

Draw.TriangleUp(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush)  
Draw.TriangleUp(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush)  
Draw.TriangleUp(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, Brush brush, bool drawOnPricePanel)  
Draw.TriangleUp(NinjaScriptBase owner, string tag, bool isAutoScale, int barsAgo, double y, Brush brush, bool drawOnPricePanel)  
Draw.TriangleUp(NinjaScriptBase owner, string tag, bool isAutoScale, DateTime time, double y, bool isGlobal, string templateName)  
Draw.TriangleUp(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)) |
| 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 triangle is tied to the chart's BarWidth and thus will scale automatically as the chart is resized |

**Examples**

| ns | |
| --- | --- |
| // Paints a red triangle pointing up on the current bar 1 tick below the low Draw.TriangleUp(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.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)) |
| 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.AndrewsPitchfork()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_andrewspitchfork.htm) >  **AndrewsPitchfork** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_andrewspitchfork.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_andrewspitchfork.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_arc.htm) |

**Definition**

Represents an object that exposes information regarding an Andrews Pitchfork [IDrawingTool](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm).

The *Standard Pitchfork* creates a trend channel out of the 3 user defined extreme price anchor points by connecting the first 2 points to form the anchor, and the next 2 points to form the retracement handle. From the first point then a trendline is drawn through the 50% midpoint of the retracement handle, parallel lines originating at the other 2 points forming the channel, while multiple further price levels could be set to allow for finer analysis.

In contrast the *Schiff Pitchfork* variant is constructed then by shifting the first anchor of the Standard Pitchfork one-half the vertical distance between the first 2 anchor points.

As further alternation the *Modified Schiff Pitchfork*variant is found by moving the first anchor to the midpoint of the original pitchfork's anchor handle, the trend-line connecting our first 2 anchor points.

**Methods and Properties**

|  |  |
| --- | --- |
| StartAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the end point of the drawing object |
| ExtensionAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the extension point of the drawing object |
| [PriceLevels](https://ninjatrader.com/es/support/helpGuides/nt8/pricelevels.htm) | A collection of prices calculated by the drawing object |
| CalculationMethod | The AndrewsPitchforkCalculationMethod property determining which method is used to calculate the pitchfork.    Possible values are:    •ModifiedSchiff  •Schiff  •StandardPitchfork |
| IsTextDisplayed | A bool value determining if the draw object should display text on the chart. |
| HandleLineStroke | A [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the handle (mid line) of the object |
| ExtensionLineStroke | A [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the equidistant trend lines of the object. |
| AnchorLineStroke | A [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the object |

**Example**

| ns | |
| --- | --- |
| // Instantiate an Andrews Pitchfork object AndrewsPitchfork myFork = Draw.AndrewsPitchfork(this, "tag1", false, 7, Low[7], 5, High[5], 1, Low[1], false, "ForkTemplate");   // Print the tag used to draw the object Print(myFork.Tag); | |
| **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.PathTool()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_pathtool().htm) >  **PathTool** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_pathtool().htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_pathtool().htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_polygon.htm) |

**Definition**

Represents an interface that exposes information regarding a PathTool [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#chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#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 PathTool object PathTool myPath = Draw.PathTool(this, "tag1", false, 20, 194, 10, 184, 13, 176, 25, 182);  // Instantiate a Path object PathTool myPathTool = Draw.PathTool(this, "tag1", false, 20, 194, 10, 184, 13, 176, 25, 182); | |
| **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.Polygon()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_polygon.htm) >  **Polygon** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_polygon.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_polygon.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_ray.htm) |

**Definition**

Represents an interface that exposes information regarding a Polyon [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#chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#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 Polygon object Polygon myPolygon = Draw.Polygon(this, "tag1", false, 20, 194, 10, 184, 13, 176, 25, 182);   // Set a new area brush for the object myPolygon.AreaBrush = Brushes.Green; | |
| **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.Ray()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_ray.htm) >  **Ray** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_ray.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_ray.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_rectangle.htm) |

**Definition**

Represents an interface that exposes information regarding a Ray [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#chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#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 Ray object Ray myRay = Draw.Ray(this, "tag1", 10, 1000, 0, 1001, Brushes.LimeGreen);   // Set a new Stroke for the object myRay.Stroke = new Stroke(Brushes.Green, DashStyleHelper.DashDot, 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.Rectangle()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_rectangle.htm) >  **Rectangle** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_rectangle.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_rectangle.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_region.htm) |

**Definition**

Represents an interface that exposes information regarding a Rectangle [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#chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the starting point of the drawing object |
| AreaBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) object representing the fill color of the draw object |
| AreaOpacity | An int value representing the opacity of the area color |
| OutlineStroke | The [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the object's outline |

**Example**

| ns | |
| --- | --- |
| // Instantiate a Rectangle object Rectangle myRec = Draw.Rectangle(this, "tag1", 10, Low[10] - TickSize, 5, High[5] + TickSize, Brushes.Blue);   // Set the object's AreaBrush to Blue myRec.AreaBrush = 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.Region()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_region.htm) >  **Region** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_region.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_region.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_regionhighlightx.htm) |

**Definition**

Represents an interface that exposes information regarding a Region [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#chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the starting point of the drawing object |
| AreaOpacity | An int value representing the opacity of the area color |
| AreaBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) object representing the fill color of the draw object |
| OutlineStroke | A Stroke used for the outline of the region |

**Example**

| ns | |
| --- | --- |
| // Instantiate a Region object Region myRegion = Draw.Region(this, "tag1", CurrentBar, 0, Bollinger(2, 14).Upper, Bollinger(2, 14).Lower, null, Brushes.Blue, 50);  // Set the object's OutlineStroke to a new Stroke myRegion.OutlineStroke = new Stroke(Brushes.Red, 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.RegionHighlightX()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_regionhighlightx.htm) >  **RegionHighlightX** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_regionhighlightx.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_regionhighlightx.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_regionhighlighty.htm) |

**Definition**

Represents an interface that exposes information regarding a Region Highlight X [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#chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the starting point of the drawing object |
| AreaBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) class representing the fill color of the draw object |
| AreaOpacity | An int value representing the opacity of the area color |
| OutlineStroke | The [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the object's outline |

**Example**

| ns | |
| --- | --- |
| // Instantiate a RegionHighlightX object RegionHighlightX myReg = Draw.RegionHighlightX(this, "tag1", 10, 0, Brushes.Blue);   // Change the object's opacity myReg.AreaOpacity = 25; | |
| **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.RegionHighlightY()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_regionhighlighty.htm) >  **RegionHighlightY** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_regionhighlighty.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_regionhighlighty.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_regressionchannel.htm) |

**Definition**

Represents an interface that exposes information regarding a Region Highlight Y [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#chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the starting point of the drawing object |
| AreaBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) class representing the fill color of the draw object |
| AreaOpacity | An int value representing the opacity of the area color |
| OutlineStroke | The [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the object's outline |

**Example**

| ns | |
| --- | --- |
| // Instantiate a RegionHighlightX object RegionHighlightY myReg = Draw.RegionHighlightY(this, "tag1", 10, 0, Brushes.Blue);   // Change the object's opacity myReg.AreaOpacity = 25; | |
| **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.RiskReward()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_riskreward.htm) >  **RiskReward** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_riskreward.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_riskreward.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_ruler.htm) |

**Definition**

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

**Methods and Properties**

|  |  |
| --- | --- |
| EntryAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the entry point of the drawing object |
| RiskAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the stop loss point of the drawing object |
| RewardAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the profit target point of the drawing object |
| Ratio | An int value determining the calculated ratio between the risk or reward based on the entry point |

**Example**

| ns | |
| --- | --- |
| // Instantiate a RiskReward object RiskReward myRR = Draw.RiskReward(this, "tag1", false, 0, High[0], 10, Low[0], 2, true);   // Change the object's risk/reward ratio to 2:1 myRR.Ratio = 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.Ruler()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_ruler.htm) >  **Ruler** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_ruler.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_ruler.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_square.htm) |

**Definition**

Represents an interface that exposes information regarding a Ruler [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#chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the end point of the drawing object |
| TextAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the text point of the drawing object |
| TextColor | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) class representing the fill color of the draw object's text area |
| LineColor | A [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the object |

**Example**

| ns | |
| --- | --- |
| // Instantiate a Ruler object Ruler myRuler = Draw.Ruler(this, "tag1", true, 4, Low[4], 3, High[3], 1, Low[1]);   // Change the object's text color to white myRuler.TextColor = Brushes.White; | |
| **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.Square()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_square.htm) >  **Square** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_square.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_square.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_text.htm) |

**Definition**

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

**Methods and Properties**

|  |  |
| --- | --- |
| Anchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the point of the drawing object |
| OutlineBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) used for the outline of the square |
| AreaBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) object representing the fill color of the draw object |

**Example**

| ns | |
| --- | --- |
| // Instantiate a Square object Square mySquare = Draw.Square(this, "tag1", true, 0, Low[0] - TickSize, Brushes.Red);   // Change the object's OutlineBrush mySquare.OutlineBrush = 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.Square()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_square.htm) >  **Square** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_square.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_square.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_text.htm) |

**Definition**

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

**Methods and Properties**

|  |  |
| --- | --- |
| Anchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the point of the drawing object |
| OutlineBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) used for the outline of the square |
| AreaBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) object representing the fill color of the draw object |

**Example**

| ns | |
| --- | --- |
| // Instantiate a Square object Square mySquare = Draw.Square(this, "tag1", true, 0, Low[0] - TickSize, Brushes.Red);   // Change the object's OutlineBrush mySquare.OutlineBrush = 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.Text()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_text.htm) >  **Text** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_text.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_text.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_textfixed.htm) |

**Definition**

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

**Methods and Properties**

|  |  |
| --- | --- |
| Anchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the point of the drawing object |
| YPixelOffset | An int value representing the offset value in pixels from within the text box area |
| Alignment | Possible values are:    TextAlignment.Center,  TextAlignment.Left,  TextAlignment.Right,  TextAlignment.Justify ([reference](https://msdn.microsoft.com/en-us/library/system.windows.textalignment(v=vs.110).aspx)) |
| AreaOpacity | An int value representing the opacity of the area color |
| AreaBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) class representing the fill color of the text box |
| Text | A string value representing the text to be drawn |
| TextBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) class representing the color of the text |
| Font | A [Font](http://msdn.microsoft.com/en-us/library/system.drawing.font_members(v=vs.90).aspx) object representing the font for the text |
| OutlineStroke | The [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to outline the text box |

**Example**

| ns | |
| --- | --- |
| // Instantiate a Text object Text myText = Draw.Text(this, "tag1", "Text to draw", 10, High[10] + (5 \* TickSize), Brushes.Black);   // Change the object's DisplayText myText.DisplayText = "New Display Text"; | |
| **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.TextFixed()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_textfixed.htm) >  **TextFixed** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_textfixed.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_textfixed.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_timecycles.htm) |

**Definition**

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

**Methods and Properties**

|  |  |
| --- | --- |
| Anchor | AAn [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the point of the drawing object |
| YPixelOffset | An int value representing the offset value in pixels from within the text box area |
| Alignment | Possible values are:    TextAlignment.Center TextAlignment.Far TextAlignment.Near  TextAlignment.Justify  ([reference](https://msdn.microsoft.com/en-us/library/system.windows.textalignment%28v=vs.110%29.aspx)) |
| AreaOpacity | An int value representing the opacity of the area color |
| AreaBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) class representing the fill color of the text box |
| DisplayText | A string value representing the text to be drawn |
| TextBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) class representing the color of the text |
| Font | A [Font](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm) object representing the font for the text |
| OutlineStroke | The [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to outline the text box |
| TextPosition | Possible values are:    TextPosition.BottomLeft  TextPosition.BottomRight  TextPosition.Center  TextPosition.TopLeft  TextPosition.TopRight |

**Example**

| ns | |
| --- | --- |
| // Instantiate a TextFixed object TextFixed myTF = Draw.TextFixed(this, "tag1", "Text to draw", TextPosition.TopRight);   // Change the object's TextPosition myTF.TextPosition = TextPosition.Center; | |
| **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.TrendChannel()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_trendchannel.htm) >  **TrendChannel** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_trendchannel.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_trendchannel.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_triangle.htm) |

**Definition**

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

**Methods and Properties**

|  |  |
| --- | --- |
| TrendStartAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the starting point of the drawing object |
| TrendEndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the end point of the drawing object |
| ParallelStartAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the starting point of the second line used in the trend channel |
| ParallelEndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the ending point of the second line used in the trend channel |
| [PriceLevels](https://ninjatrader.com/es/support/helpGuides/nt8/pricelevels.htm) | A collection of prices calculated by the drawing object |

**Example**

| ns | |
| --- | --- |
| // Instantiate a TrendChannel object TrendChannel myTC = Draw.TrendChannel(this, "tag1", true, 10, Low[10], 0, High[0], 10, High[10] + 5 \* TickSize);             // Increase the y-axis position of the object's TrendEndAnchor myTC.TrendEndAnchor.Price += 15; | |
| **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.Triangle()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_triangle.htm) >  **Triangle** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_triangle.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_triangle.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_triangledown.htm) |

**Definition**

Represents an interface that exposes information regarding a Triangle [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#chartanchor) representing the starting point of the drawing object |
| MiddleAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the middle point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the starting point of the drawing object |
| AreaBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) class representing the fill color of the draw object |
| AreaOpacity | An int value representing the opacity of the area color |
| OutlineStroke | The [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the object's outline |

**Example**

| ns | |
| --- | --- |
| // Instantiate a Triangle object Triangle myTri = Draw.Triangle(this, "tag1", 4, Low[4], 3, High[3], 1, Low[1], Brushes.Blue);               // Change the object's AreaOpacity myTri.AreaOpacity = 100; | |
| **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.TriangleDown()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_triangledown.htm) >  **TriangleDown** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_triangledown.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_triangledown.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_triangleup.htm) |

**Definition**

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

**Methods and Properties**

|  |  |
| --- | --- |
| Anchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the point of the drawing object |
| AreaBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) class representing the fill color of the draw object |
| OutlineBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) class representing the outline color of the draw object |

**Example**

| ns | |
| --- | --- |
| // Instantiate a TriangleDown object TriangleDown myTri = Draw.TriangleDown(this, "tag1", true, 0, Low[0] - TickSize, Brushes.Red);               // Change the object's AreaBrush myTri.AreaBrush = Brushes.Beige; | |
| **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.TriangleUp()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_triangleup.htm) >  **TriangleUp** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_triangleup.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_triangleup.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_verticalline.htm) |

**Definition**

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

**Methods and Properties**

|  |  |
| --- | --- |
| Anchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) representing the point of the drawing object |
| AreaBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) class representing the fill color of the draw object |
| OutlineBrush | A [Brush](http://msdn.microsoft.com/en-us/library/system.windows.media.brush(v=vs.110).aspx) class representing the outline color of the draw object |

**Examples**

| ns | |
| --- | --- |
| // Instantiate a TriangleUp object TriangleUp myTri = Draw.TriangleUp(this, "tag1", true, 0, Low[0] - TickSize, Brushes.Red);               // Change the object's AreaBrush myTri.AreaBrush = Brushes.Beige; | |
| **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#chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#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) >  **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) >  **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) object that represents the color of the current chart bar.

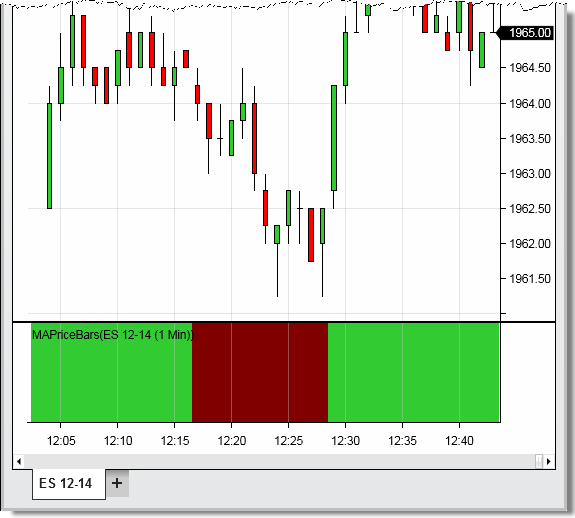
**Syntax**

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) >  **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) 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) 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) 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) > [Educational Resources](https://ninjatrader.com/es/support/helpGuides/nt8/educational_resources.htm) >  **Working with Brushes** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/educational_resources.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_chart_object_coordinates.htm) |

In order to achieve custom rendering for various chart related objects, a Brush is used to "paint" an area or another chart object.  There are a number of different brushes which are available through the .NET Framework, where the most common type of brush is a [SolidColorBrush](https://msdn.microsoft.com/en-us/library/system.windows.media.solidcolorbrush(v=vs.110).aspx) which is used to paint an area with a single solid color.

|  |
| --- |
| **Notes**:  The following document is written in sequential fashion, starting with the most simple concepts, to the more advance topics.  The majority of the brushes discussed in this document will be referred to as "**WPF" brushes** which exist in the System.Windows.Media namespace, however there are also **"SharpDX" brushes** which exist in the 3rd party SharpDX.Direct2D1 nampspace used for advanced chart rendering.  Advanced brush types should **ONLY** be used by experienced programmers familiar with .NET graphics functionality. |

tog_minus        [Understanding predefined brushes](javascript:HMToggle('toggle','Understandingpredefinedbrushes','Understandingpredefinedbrushes_ICON'))

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Using Predefined Brushes**  For convenience, the .NET Framework supplies a collection of static predefined Brushes, such as Red or Green.  The advantage to using these brushes is that they are readily available, properly named to quickly find a simple color value, and can be reused on-the-fly without having to recreate an instance of the brush at run time, and do not need to be otherwise managed.  There are 256 predefined named brushes which are available in the Brushes class.  You can browse this list in the NinjaScript editor just by typing Brushes. and using Intelliprompt to find the desired named brush of your choice.     |  | | --- | | **Note**:   Since predefined brushes are static, properties of the brush object (such as Color, Opacity, etc.) **CANNOT** be modified.  However, this also means predefined brushes are thread-safe and do **NOT** need to be frozen.  For customizing and freezing a brush, please see the section below on *Creating a Custom Solid Color Brush*. |       Brushes       |  | | --- | | **Tip**:  You can also find a list of these predefined brushes as well as their hexadecimal value on the MSDN article for the [Brushes Class](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes(v=vs.110).aspx) |        | ns |  | | --- | --- | | // set the chart's background color to a predefined "Blue" brush BackBrush = Brushes.Blue;   //draw a line using a predefined "LimeGreen" brush. Draw.Line(this, "tag1", false, 10, 1000, 0, 1001, Brushes.LimeGreen, DashStyleHelper.Dot, 2); | | |

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

tog_minus        [Understanding custom brushes](javascript:HMToggle('toggle','Understandingcustombrushes','Understandingcustombrushes_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Creating a Custom Solid Color Brush**  In cases where you would like more specific color than one of the predefined brushes, you can optionally create your own **Brush** object to be used for custom rendering.  In order to achieve this, you will need to initiate your own custom brush object, where you can then specify your color using RGB (red, green, blue) values [Color.FromRgb()](https://msdn.microsoft.com/en-us/library/system.windows.media.color.fromrgb(v=vs.110).aspx).     |  | | --- | | **Notes**:  •Anytime you create a custom brush that will be used by NinjaTrader rendering it must be frozen using the .[Freeze()](https://msdn.microsoft.com/en-us/library/ms557735(v=vs.110).aspx)  method due to the multi-threaded nature of NinjaTrader.  •You may have up to 65535 unique Brush instances, therefore, using static predefined brushes (as in the section above) 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. |        | ns |  | | --- | --- | | // initiate new solid color brush with custom blue color Brush myBrush = new SolidColorBrush(Color.FromRgb(56, 120, 153)); myBrush.Freeze();   Draw.Line(this, "tag1", true, 10, 1000, 0, 1001, myBrush, DashStyleHelper.Dot, 2); | |      |  | | --- | | **Warning**:  If you do not call .[Freeze()](https://msdn.microsoft.com/en-us/library/ms557735(v=vs.110).aspx) on a custom defined brush **WILL**eventually result in threading errors should you try to modify or access that brush after it is defined. |       **Creating a Transparent Solid Color Brush**  You can create a transparent brush using the [Color.FromArgb()](https://msdn.microsoft.com/en-us/library/system.windows.media.color.fromargb(v=vs.110).aspx) where the A parameter defines alpha transparency.     |  | | --- | | **Note**:   Anytime you create a custom brush that will be used by NinjaTrader rendering it must be frozen using the .[Freeze()](https://msdn.microsoft.com/en-us/library/ms557735(v=vs.110).aspx)  method due to the multi-threaded nature of NinjaTrader. |        | ns |  | | --- | --- | | // initiate new solid color brush which has an alpha (transparency) value of 100 MyBrush = new SolidColorBrush(Color.FromArgb(100, 56, 120, 153)); myBrush.Freeze();   Draw.Line(this, "tag1", true, 10, 1000, 0, 1001, myBrush, DashStyleHelper.Dot, 2); | |        |  | | --- | | **Warning**:  If you do not call .[Freeze()](https://msdn.microsoft.com/en-us/library/ms557735(v=vs.110).aspx) on a custom defined brush **WILL**eventually result in threading errors should you try to modify or access that brush after it is defined. | |

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

tog_minus        [Using brushes defined on the user interface](javascript:HMToggle('toggle','Userdefinedbrushes','Userdefinedbrushes_ICON'))

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Saving a Brush as a user defined property (Serialization)**  If you would like a brush to become a public UI property, meaning the brush can be set up and defined by a user during configuration, it is important to be able to save the user's brush selection in order to restore that brush either from a workspace or from a template file at a later time.  Saving a custom defined user input is done through a concept of [Serialization](https://msdn.microsoft.com/en-us/library/ms233843.aspx) which writes the object and its value to a .xml file.  This process normally works fine for a simple user defined value type (such as a double or an int) but for more complex types such as Brushes, the object itself cannot be serialized directly to the .xml file and will result in errors upon saving the indicator or strategy to a workspace or template file.  The example below will demonstrate and explain how to properly store a user define brush input which will be correctly serialized.    In order to achieve the desired behavior of saving the user defined brush input, we will add the [XmlIgnore](https://msdn.microsoft.com/en-us/library/system.xml.serialization.xmlignoreattribute(v=vs.110).aspx) property attribute to the public brush resource, which essentially tells the serialization routine to ignore this property.     | ns | | --- | | [XmlIgnore] public Brush MyBrush { get; set; } |       In its place, we create a new public string called "MyBrushSerialize" which will convert the public "MyBrush" to a string type which can then be processed by the serialization routines.  We also add the [Browsable(false)](https://msdn.microsoft.com/en-us/library/system.componentmodel.browsableattribute(v=vs.110).aspx) attribute to this public string to prevent this property from showing up on the UI, which is of no value to the end user.     | ns | | --- | | [Browsable(false)] public string MyBrushSerialize {   get { return Serialize.BrushToString(MyBrush); }   set { MyBrush = Serialize.StringToBrush(value); } } |        |  | | --- | | **Tip**: For a complete example of **User Definable Color Inputs**, please see the reference sample [here](https://ninjatrader.com/es/support/helpGuides/nt8/user_definable_color_inputs.htm). |       **Adding a User Defined Brush to the Color Picker**  You can optionally define a custom brush to be added to the standard color picker by using a [CustomBrush] attribute to a public brush.  The CustomBrush attribute will then add it to the color picker menu for that indicator when you look through the plots, lines, or other brushes from the indicators configured menu and will be listed toward the top of the list (as pictured below)     | ns | | --- | | [CustomBrush] public Brush MyBrush {   get { return new SolidColorBrush(Color.FromRgb(25, 175, 185)); }   set { } } |     custom_brush |

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

tog_minus        [Using advanced brush types (SharpDX)](javascript:HMToggle('toggle','AdvancedBrushTypesSharpDX','AdvancedBrushTypesSharpDX_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Understanding SharpDX Brushes**  While the majority of the NinjaTrader platform's UI is **WPF**, under the hood, chart's use a **DirectX API** for faster performance.  To render custom objects to a chart during [OnRender()](https://ninjatrader.com/es/support/helpGuides/nt8/onrender.htm), a particular **SharpDX** **Brush** object must be implemented which reside in the **SharpDX.Direct2D1** namespace.   These brushes can then be passed as arguments to the **SharpDX** [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/rendertarget.htm) methods such [FillRectangle()](https://ninjatrader.com/es/support/helpGuides/nt8/fillrectangle.htm), [DrawLine()](https://ninjatrader.com/es/support/helpGuides/nt8/drawline2.htm), etc.  While **SharpDX Brushes** behave much the same as previously discussed **WPF** **Brushes**, there are a few special considerations you must take as detailed in the following sections.     |  | | --- | | **Note**:  The **SharpDX Brushes** used in [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/rendertarget.htm) methods should **NOT** be confused with the **WPF Brushes** used with [DrawingTool Draw](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) methods. |       **Creating a SharpDX Brush**  A [SharpDX Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) must be created either in **OnRender()** or **RenderTargetChanged()**.  If you have custom brushes which may be changed on various conditions such as in OnBarUpdate() or by a user during OnStateChange(), or you are pre-computing a custom brush for performance optimization, you will need to ensure the actual SharpDX instance is updated in OnRender() or RenderTargetChange().     |  | | --- | | **Warning**:  Each DirectX render target requires its own brushes. You **MUST** create brushes directly in **OnRender()** or using **OnRenderTargetChanged()**.  If you do not you will receive an error at runtime similar to:   ***"A direct X error has occured while rendering the chart: HRESULT: [0x88990015], Module: [SharpDX.Direct2D1], ApiCode: [D2DERR\_WRONG\_RESOURCE\_DOMAIN/WrongResourceDomain], Message: The resource was realized on the wrong render target. : Each DirectX render target requires its own brushes. You must create brushes directly in OnRender() or using OnRenderTargetChanged().***    Please see [OnRenderTargetChanged()](https://ninjatrader.com/es/support/helpGuides/nt8/onrendertargetchanged.htm) for examples of a brush that needs to be recalculated, or [OnRender()](https://ninjatrader.com/es/support/helpGuides/nt8/onrender.htm) for an example of recreating a static brush. |        | ns | | --- | | // use predefined "Blue" SharpDX Color SharpDX.Direct2D1.SolidColorBrush solidBlueDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.Blue);   // create custom Brush using a "Red" SharpDX Color with "Alpha" (0.100f) transparency/opacity SharpDX.Direct2D1.SolidColorBrush transparentRedDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, new SharpDX.Color4(new SharpDX.Color3(220f, 0f, 0f), 0.100f)); |       **Converting to SharpDX Brush**  For convenience, you can convert a computed WPF Brush to a [SharpDX Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) using the [ToDxBrush(](https://ninjatrader.com/es/support/helpGuides/nt8/dxextensions_todxbrush.htm)) extension method.     |  | | --- | | **Warning**:  Converting **ToDxBrush()** can result in performance issues depending on the number of brushes being used. If you experience performance issues with your custom **SharpDX** rendering, you should favor using **SharpDX** brushes directly instead of converting the brush using **ToDxBrush().** |        | ns | | --- | | // convert predefined WPF "Blue" to SharpDX Brush SharpDX.Direct2D1.Brush blueDXBrush = Brushes.Blue.ToDxBrush(RenderTarget);   // convert the computed WPF Brush to SharpDX Brush SharpDX.Direct2D1.Brush customDXBrush = customWPFBrush.ToDxBrush(RenderTarget); |       **Disposing DXBrush**  Since **SharpDX Brushes** reference unmanaged resources, these brushes should always be [disposed](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase.htm) of after they have been used.     |  | | --- | | **Warning**:  Failing to dispose of a [SharpDX Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) and other unmanaged resources can cause the platform to utilize more memory than necessary. |        | ns | | --- | | customDXBrush.Dipose(); |     **Using Complex Brushes**  In addition to the [SolidColorBrush](https://msdn.microsoft.com/en-us/library/system.windows.media.solidcolorbrush(v=vs.110).aspx) object demonstrated on this page, the .NET Framework provides more complex brushes which have more attributes than just filling an area with a solid color.  Information on these special types of brushes can be found on the MSDN website: [LinearGradientBrush](https://msdn.microsoft.com/en-us/library/system.windows.media.lineargradientbrush(v=vs.110).aspx), [RadialGradientBrush](https://msdn.microsoft.com/en-us/library/system.windows.media.radialgradientbrush(v=vs.110).aspx), [ImageBrush](https://msdn.microsoft.com/en-us/library/system.windows.media.imagebrush(v=vs.110).aspx).    These complex types also have an equivalent found in the**SharpDX SDK Reference**: [SharpDX.Direct2D1.LinearGradientBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_lineargradientbrush.htm), [SharpDX.Direct2D1.RadialGradientBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_radialgradientbrush.htm) |

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

|  |  |
| --- | --- |
| **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) 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) > [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) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **IDrawingTool** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawingtools_drawobjects.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/pricelevels.htm) |

**Definition**

Represents an interface that exposes information regarding a drawn chart object.

IDrawingTool Properties are standard properties that are shared by all drawing tools.

Each specific **IDrawingTool** will have its own uniquely named **ChartAnchor**representing where the object was drawn on the chart.  The name and number of **ChartAnchors** will be specific to that drawing tool (e.g., StartAnchor, EndAnchor, etc), however the fields available will be the same (e.g., BarsAgo, DrawnOnBar, etc).  Details on those shared fields are outlined in the **ChartAnchor Properties** section toward the bottom of this topic.

|  |
| --- |
| **Note**:  For implementing a custom Drawing Tool project, please see the [DrawingTools](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) section of this help guide. |

**IDrawingTool Properties**

|  |  |
| --- | --- |
| Anchors | A read-only collection of all of the [IDrawingTool's ChartAnchors](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) |
| AttachedTo | An enum determining where the drawing tool is attached.    Possible values are:  •AttachedToType.Bars,  •AttachedToType.GlobalInstrument,  •AttachedToType.Indicator,  •AttachedToType.Strategy |
| DrawingState | The current [DrawingState](https://ninjatrader.com/es/support/helpGuides/nt8/drawingstate.htm) of the drawing tool |
| DrawnBy | An object value indicating which type of NinjaScript the drawing tool originated (null if user drawn) |
| IsAttachedToNinjaScript | A read-only bool indicating if the drawing tool is attached to an indicator or strategy |
| IgnoresUserInput | A read-only bool determining if the drawing tool can be interacted with by the user. |
| IsGlobalDrawingTool | A bool determining if the drawing tool displays on all charts of the instrument |
| IsLocked | A bool determining if the drawing tool can be moved |
| IsSeparateZOrder | A bool determining if the drawing tool will reside on a different ZOrder from the NinjaScript object it was drawn |
| IsUserDrawn | A read-only bool indicating if drawing tool was manually drawn by a user |
| PanelIndex | An int value representing the panel the drawing tool resides |
| SupportsAlerts | A read-only bool indicating if the drawing tool can be used for creating an alert |
| Tag | A string value representing the unique ID of the draw object. (Global draw objects will have an "@" added as a prefix to the string) |
| ZOrderType | A read-only enum indicating the order in which the drawing tool will be drawn.    Possible values are:  •DrawingToolZOrder.Normal,  •DrawingToolZOrder.AlwaysDrawnFirst,  •DrawingToolZOrder.AlwaysDrawnLast |

**ChartAnchor Properties**

|  |  |
| --- | --- |
| <ChartAnchor>.BarsAgo | An int representing the "barsAgo" value that was passed to the Draw method    **Note**:  This value will **NOT** be set for objects drawn manually |
| <ChartAnchor>.DisplayName | A string representing name of the DrawingTool's chart anchor that is displaying on the UI |
| <ChartAnchor>.DrawingTool | The IDrawingTool object which created the DrawingTool's chart anchor object |
| <ChartAnchor>.DrawnOnBar | An int representing the [CurrentBar](https://ninjatrader.com/es/support/helpGuides/nt8/currentbar.htm) value that the DrawingTool's chart anchor was drawn |
| <ChartAnchor>.IsNinjaScriptDrawn | A bool indicating the object was drawn programmatically |
| <ChartAnchor>.Price | A double representing the price the  DrawingTool's chart anchor was drawn |
| <ChartAnchor>.SlotIndex | A double representing the DrawingTool's chart anchor index value the anchor was drawn |
| <ChartAnchor>.Time | A DateTime representing the time value the DrawingTool's chart anchor was drawn |

**Examples**

| ns | |
| --- | --- |
| Text myText; protected override void OnBarUpdate() {     if(CurrentBar == 50)     myText = Draw.Text(this, "tag", "test", 0, High[0]);         if(myText != null)   {               Print(myText.Anchor.DrawnOnBar); // drawn on bar 50   }   } | |
| **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) >  **SimpleFont** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/istradingdaydefined.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_applyto.htm) |

**Definition**

Defines a particular font configuration.

|  |
| --- |
| **Note**: **SimpleFont** objects are used for various [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) methods, and can be used when defining UI element for Add-ons. |

**Constructors**

|  |  |
| --- | --- |
| SimpleFont() | Creates a **SimpleFont** object using a family name of "Arial" and a size of "12" |
| SimpleFont(string familyName, int size) | Creates a **SimpleFont** object using the specified family name and size |

**Methods and Properties**

|  |  |
| --- | --- |
| Bold | A bool value determining if the the Font is bold style |
| Family | A [FontFamily](https://msdn.microsoft.com/en-us/library/system.windows.media.fontfamily(v=vs.110).aspx) representing a family of Fonts |
| Italic | A bool value determining if the the Font is italic style |
| Size | A double value determining the size of font in WPF units (please see the tip below) |
| Typeface | A [Typeface](https://msdn.microsoft.com/en-us/library/system.windows.media.typeface%28v=vs.110%29.aspx) used to represent the variation of the font used |
| [ApplyTo()](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_applyto.htm) | Applies a custom [SimpleFont](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm) object's properties (family, size, and style) to a [Windows Control](https://msdn.microsoft.com/en-us/library/system.windows.controls.control(v=vs.110).aspx) |
| [ToDirectWriteTextFormat()](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_todirectwritetextformat.htm) | Converts a SimpleFont object to a SharpDX compatible font which can be used for chart rendering. |

|  |
| --- |
| **Tip**: The WPF unit used is the default px one, so device independent pixels. With a default system DPI setting of 96, the physical pixel on the screen would be identical in size, but can vary if a custom DPI is employed.  Both should not be confused with the points based font sizing known from other familiar Windows applications like Word, the advantage here is that the non points based size measurement will increase / decrease in size if the system DPI is changed - a more detailed discussion is located [here](https://blogs.msdn.microsoft.com/text/2009/12/11/wpf-text-measurement-units/). |

**Examples**

| ns | |
| --- | --- |
| // create custom Courier New, make it big and bold NinjaTrader.Gui.Tools.SimpleFont myFont = new NinjaTrader.Gui.Tools.SimpleFont("Courier New", 12) { Size = 50, Bold = true };   Draw.Text(this, "myTag", false, "Hi There!", 0, Low[0], 5, Brushes.Blue, myFont, TextAlignment.Center, Brushes.Black, null, 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) > [SimpleFont](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm) >  **ApplyTo()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_todirectwritetextformat.htm) |

**Definition**

Applies a custom [SimpleFont](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm) object's properties (family, size, and style) to a [Windows Control](https://msdn.microsoft.com/en-us/library/system.windows.controls.control(v=vs.110).aspx)

**Method Return Value**

This method does not return a value.

**Syntax**

<SimpleFont>.ApplyTo(DependencyObject target)

|  |  |
| --- | --- |
| target | The [DependencyObject](https://msdn.microsoft.com/en-us/library/system.windows.dependencyobject(v=vs.110).aspx) to apply the SimpleFont object |

**Examples**

| ns | |
| --- | --- |
| // Define the custom button control object System.Windows.Controls.Button myButton = new System.Windows.Controls.Button {   Name = "myButton",   Content = "Buy",   Foreground = Brushes.White,   Background = Brushes.Green, };   // Create a custom SimpleFont object and then apply it to the button SimpleFont myFont = new SimpleFont("Consolas", 22);   myFont.ApplyTo(myButton); | |
| **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) > [SimpleFont](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm) >  **ToDirectWriteTextFormat()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_applyto.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/indicators.htm) |

**Definition**

Converts a [SimpleFont](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm) object to a [SharpDX](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx.htm) compatible font which can be used for chart rendering.

|  |
| --- |
| **Note**:  For more information please see the educational resource on [Using SharpDX for Custom Chart Rendering](https://ninjatrader.com/es/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm) |

**Method Return Value**

A [DirectWrite.TextFormat](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textformat.htm) object

|  |
| --- |
| **Warning**:  Thereturned**DirectWrite.TextFormat** object should be disposed of immediately when finished drawing text. |

**Syntax**

<SimpleFont>.ToDirectWriteTextFormat()

**Examples**

| ns | |
| --- | --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // Set text to chart label simple font object   SharpDX.DirectWrite.TextFormat textFormat = chartControl.Properties.LabelFont.ToDirectWriteTextFormat();     // use the textFormat in a RenderTarget.DrawText() or DrawTextLayout() method     // do not forget to dispose text format when finished   textFormat.Dispose(); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Editor](https://ninjatrader.com/es/support/helpGuides/nt8/editor.htm) >  **Code Snippets** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ns_wizard.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/editor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/compile_errors.htm) |

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

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

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

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

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

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

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

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

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

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

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Educational Resources](https://ninjatrader.com/es/support/helpGuides/nt8/educational_resources.htm) > [Reference Samples](https://ninjatrader.com/es/support/helpGuides/nt8/reference_samples.htm) > [Indicator](https://ninjatrader.com/es/support/helpGuides/nt8/indicator2.htm) >  **Getting indicator values from a specified time** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/exposing_indicator_values_that.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/indicator2.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/manipulating_datetime_objects.htm) |

Sometimes, you may want to access a value from a historical point in time, but have not kept track of the value to make this readily available. With NinjaScript, it is possible to pick a bar based on time to access that value. GetBar() returns the number of bars ago that holds the same timestamp of the time you request. This sample demonstrates how to get an indicator value from 9:30AM of the previous trading day.

**Key concepts in this example**

•Obtaining a Simple Moving Average value from a specific time by referencing the bar number for that time.

**Important related documentation**

•[GetBar()](https://ninjatrader.com/es/support/helpGuides/nt8/getbar.htm)

•[Draw.Line()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_line.htm)

•[Time](https://ninjatrader.com/es/support/helpGuides/nt8/iseries_time.htm)

•[Sessions](https://ninjatrader.com/es/support/helpGuides/nt8/tradinghours_sessions.htm)

•[DateTime](https://msdn.microsoft.com/en-us/library/system.datetime(v=vs.110).aspx)

**Import instructions**

1.Download the file contained in this Help Guide topic to your PC desktop

2.From the Control Center window, select the menu Tools > Import > NinjaScript

3.Select the downloaded file

[SampleGetBar\_NT8.zip](https://ninjatrader.com/support/helpGuides/nt8/samples/SampleGetBar_NT8.zip)

//

// Copyright (C) 2016, NinjaTrader LLC <www.ninjatrader.com>.

// NinjaTrader reserves the right to modify or overwrite this NinjaScript component with each release.

//

#region Using declarations

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using System.Windows.Media;

using System.Xml.Serialization;

using NinjaTrader.Cbi;

using NinjaTrader.Gui;

using NinjaTrader.Gui.Chart;

using NinjaTrader.Gui.SuperDom;

using NinjaTrader.Data;

using NinjaTrader.NinjaScript;

using NinjaTrader.Core.FloatingPoint;

using NinjaTrader.NinjaScript.DrawingTools;

#endregion

// This namespace holds indicators in this folder and is required. Do not change it.

namespace NinjaTrader.NinjaScript.Indicators

{

/// <summary>

/// The SMA (Simple Moving Average) is an indicator that shows the average value of a security's price over a period of time.

/// </summary>

public class SMA : Indicator

{

private double priorSum;

private double sum;

protected override void OnStateChange()

{

if (State == State.SetDefaults)

{

Description = NinjaTrader.Custom.Resource.NinjaScriptIndicatorDescriptionSMA;

Name = NinjaTrader.Custom.Resource.NinjaScriptIndicatorNameSMA;

IsOverlay = true;

IsSuspendedWhileInactive = true;

Period = 14;

AddPlot(Brushes.Goldenrod, NinjaTrader.Custom.Resource.NinjaScriptIndicatorNameSMA);

}

else if (State == State.Configure)

{

priorSum = 0;

sum = 0;

}

}

protected override void OnBarUpdate()

{

if (BarsArray[0].BarsType.IsRemoveLastBarSupported)

{

if (CurrentBar == 0)

Value[0] = Input[0];

else

{

double last = Value[1] \* Math.Min(CurrentBar, Period);

if (CurrentBar >= Period)

Value[0] = (last + Input[0] - Input[Period]) / Math.Min(CurrentBar, Period);

else

Value[0] = ((last + Input[0]) / (Math.Min(CurrentBar, Period) + 1));

}

}

else

{

if (IsFirstTickOfBar)

priorSum = sum;

sum = priorSum + Input[0] - (CurrentBar >= Period ? Input[Period] : 0);

Value[0] = sum / (CurrentBar < Period ? CurrentBar + 1 : Period);

}

}

#region Properties

[Range(1, int.MaxValue), NinjaScriptProperty]

[Display(ResourceType = typeof(Custom.Resource), Name = "Period", GroupName = "NinjaScriptParameters", Order = 0)]

public int Period

{ get; set; }

#endregion

}

}

#region NinjaScript generated code. Neither change nor remove.

namespace NinjaTrader.NinjaScript.Indicators

{

public partial class Indicator : NinjaTrader.Gui.NinjaScript.IndicatorRenderBase

{

private SMA[] cacheSMA;

public SMA SMA(int period)

{

return SMA(Input, period);

}

public SMA SMA(ISeries<double> input, int period)

{

if (cacheSMA != null)

for (int idx = 0; idx < cacheSMA.Length; idx++)

if (cacheSMA[idx] != null && cacheSMA[idx].Period == period && cacheSMA[idx].EqualsInput(input))

return cacheSMA[idx];

return CacheIndicator<SMA>(new SMA(){ Period = period }, input, ref cacheSMA);

}

}

}

namespace NinjaTrader.NinjaScript.MarketAnalyzerColumns

{

public partial class MarketAnalyzerColumn : MarketAnalyzerColumnBase

{

public Indicators.SMA SMA(int period)

{

return indicator.SMA(Input, period);

}

public Indicators.SMA SMA(ISeries<double> input , int period)

{

return indicator.SMA(input, period);

}

}

}

namespace NinjaTrader.NinjaScript.Strategies

{

public partial class Strategy : NinjaTrader.Gui.NinjaScript.StrategyRenderBase

{

public Indicators.SMA SMA(int period)

{

return indicator.SMA(Input, period);

}

public Indicators.SMA SMA(ISeries<double> input , int period)

{

return indicator.SMA(input, period);

}

}

}

#endregion//

// Copyright (C) 2015, NinjaTrader LLC <www.ninjatrader.com>.

// NinjaTrader reserves the right to modify or overwrite this NinjaScript component with each release.

//

#region Using declarations

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using System.Windows.Media;

using System.Xml.Serialization;

using NinjaTrader.Cbi;

using NinjaTrader.Gui;

using NinjaTrader.Gui.Chart;

using NinjaTrader.Gui.SuperDom;

using NinjaTrader.Data;

using NinjaTrader.NinjaScript;

using NinjaTrader.Core.FloatingPoint;

using NinjaTrader.NinjaScript.DrawingTools;

#endregion

// This namespace holds all indicators and is required. Do not change it.

namespace NinjaTrader.NinjaScript.Indicators

{

public class SampleGetBar : Indicator

{

private int sMAPeriod; // Default setting for sMAPeriod

private int barsAgo; // Variable to hold a bars ago value.

private DateTime timeOfInterest; // DateTime object to hold the time we're interested in.

protected override void OnStateChange()

{

if(State == State.SetDefaults)

{

Description = @"Sample Getting a value from an indicator for a specific time";

Name = "Sample GetBar";

Calculate = Calculate.OnBarClose;

IsOverlay = true;

DisplayInDataBox = true;

DrawOnPricePanel = true;

DrawHorizontalGridLines = true;

DrawVerticalGridLines = true;

PaintPriceMarkers = true;

ScaleJustification = NinjaTrader.Gui.Chart.ScaleJustification.Right;

sMAPeriod = 14;

barsAgo = 0;

AddPlot(Brushes.Orange, "SMAPlot");

}

}

protected override void OnBarUpdate()

{

// Plots the SMA indicator.

SMAPlot[0] = SMA(sMAPeriod)[0];

/\* This sample assumes instrument sessions that go from Monday-Friday and not the weekends.

If you are using an instrument that has weekend trading sessions like the ES futures, please be aware that the logic presented will need to be

modified to reflect the trading sessions for your particular instrument.

Here, timeOfInterest is reset once per day at session break. \*/

if (Bars.IsFirstBarOfSession && IsFirstTickOfBar)

{

DateTime holdDT = new DateTime(Time[0].Year, Time[0].Month, Time[0].Day, 9, 30, 0);

/\* If the day is Monday, and we want the value from Friday, we must subtract 3 days from the current date.

If the day is Tuesday-Friday, just subtract one day. \*/

if (Time[0].DayOfWeek == DayOfWeek.Monday)

timeOfInterest = holdDT.AddDays(-3);

else

timeOfInterest = holdDT.AddDays(-1);

}

/\* Determine the number of bars ago it would take to access the 9:30AM bar for the previous trading day by pulling the bars index and subtracting from the CurrentBar.

GetBar() returns returns the first bar that matches the time stamp for the 9:30AM bar if it exists or the number for the first bar after 9:30AM

if there isn t a specific bar with the exact 9:30AM timestamp (e.g. tick-based bars). With tick bars, it is possible

for many bars to have the same exact time stamp. In this case, the first bar of all the bars with the same timestamp is returned.\*/

barsAgo = CurrentBar - Bars.GetBar(timeOfInterest);

/\* If GetBar returns 0, the time you are looking for would either be the current bar or at some time after the current bar.

Also, make sure there are enough bars to draw the line 10 bars back. \*/

if (barsAgo != 0 && CurrentBar > 10)

{

// Draw a line with the value of the SMA the correct number of bars ago.

Draw.Line(this, "yesterday SMA", true, 10, SMAPlot[barsAgo], 0, SMAPlot[barsAgo], Brushes.Red, DashStyleHelper.Solid, 5);

//DrawLine("yesterday SMA", false, 10, SMAPlot[barsAgo], 0, SMAPlot[barsAgo], Color.Black, DashStyle.Solid, 2);

}

}

#region Properties

[Browsable(false)]

[XmlIgnore()]

public Series<double> SMAPlot

{

get { return Values[0]; }

}

[Range(1, int.MaxValue), NinjaScriptProperty]

[Display(ResourceType = typeof(Custom.Resource), Name = "SMAPeriod", GroupName = "NinjaScriptParameters", Order = 0)]

public int SMAPeriod

{ get { return sMAPeriod;} set{ sMAPeriod = value;} }

#endregion

}

}

#region NinjaScript generated code. Neither change nor remove.

namespace NinjaTrader.NinjaScript.Indicators

{

public partial class Indicator : NinjaTrader.Gui.NinjaScript.IndicatorRenderBase

{

private SampleGetBar[] cacheSampleGetBar;

public SampleGetBar SampleGetBar(int sMAPeriod)

{

return SampleGetBar(Input, sMAPeriod);

}

public SampleGetBar SampleGetBar(ISeries<double> input, int sMAPeriod)

{

if (cacheSampleGetBar != null)

for (int idx = 0; idx < cacheSampleGetBar.Length; idx++)

if (cacheSampleGetBar[idx] != null && cacheSampleGetBar[idx].SMAPeriod == sMAPeriod && cacheSampleGetBar[idx].EqualsInput(input))

return cacheSampleGetBar[idx];

return CacheIndicator<SampleGetBar>(new SampleGetBar(){ SMAPeriod = sMAPeriod }, input, ref cacheSampleGetBar);

}

}

}

namespace NinjaTrader.NinjaScript.MarketAnalyzerColumns

{

public partial class MarketAnalyzerColumn : MarketAnalyzerColumnBase

{

public Indicators.SampleGetBar SampleGetBar(int sMAPeriod)

{

return indicator.SampleGetBar(Input, sMAPeriod);

}

public Indicators.SampleGetBar SampleGetBar(ISeries<double> input , int sMAPeriod)

{

return indicator.SampleGetBar(input, sMAPeriod);

}

}

}

namespace NinjaTrader.NinjaScript.Strategies

{

public partial class Strategy : NinjaTrader.Gui.NinjaScript.StrategyRenderBase

{

public Indicators.SampleGetBar SampleGetBar(int sMAPeriod)

{

return indicator.SampleGetBar(Input, sMAPeriod);

}

public Indicators.SampleGetBar SampleGetBar(ISeries<double> input , int sMAPeriod)

{

return indicator.SampleGetBar(input, sMAPeriod);

}

}

}

#endregion

|  |  |
| --- | --- |
| **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.Line()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_line.htm) >  **Line** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_line.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_line.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_pathtool().htm) |

**Definition**

Represents an interface that exposes information regarding a 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#chartanchor) representing the starting point of the drawing object |
| EndAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#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 Line object NinjaTrader.NinjaScript.DrawingTools.Line myLine = Draw.Line(this, "tag1", false, 10, 1000, 0, 1001, Brushes.LimeGreen, DashStyleHelper.Dot, 2);   // Set a new Stroke for the object myLine.Stroke = new Stroke(Brushes.Green, DashStyleHelper.Dash, 5); |

|  |  |
| --- | --- |
| **Note**: To differentiate between NinjaTrader.NinjaScript.DrawingTools.Line and NinjaTrader.Gui.Line when assigning a Line object, you will need to invoke the former path explicitly, as seen in the example above. | |
| **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) >  **IDrawingTool** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawingtools_drawobjects.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/pricelevels.htm) |

**Definition**

Represents an interface that exposes information regarding a drawn chart object.

IDrawingTool Properties are standard properties that are shared by all drawing tools.

Each specific **IDrawingTool** will have its own uniquely named **ChartAnchor**representing where the object was drawn on the chart.  The name and number of **ChartAnchors** will be specific to that drawing tool (e.g., StartAnchor, EndAnchor, etc), however the fields available will be the same (e.g., BarsAgo, DrawnOnBar, etc).  Details on those shared fields are outlined in the **ChartAnchor Properties** section toward the bottom of this topic.

|  |
| --- |
| **Note**:  For implementing a custom Drawing Tool project, please see the [DrawingTools](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) section of this help guide. |

**IDrawingTool Properties**

|  |  |
| --- | --- |
| Anchors | A read-only collection of all of the [IDrawingTool's ChartAnchors](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) |
| AttachedTo | An enum determining where the drawing tool is attached.    Possible values are:  •AttachedToType.Bars,  •AttachedToType.GlobalInstrument,  •AttachedToType.Indicator,  •AttachedToType.Strategy |
| DrawingState | The current [DrawingState](https://ninjatrader.com/es/support/helpGuides/nt8/drawingstate.htm) of the drawing tool |
| DrawnBy | An object value indicating which type of NinjaScript the drawing tool originated (null if user drawn) |
| IsAttachedToNinjaScript | A read-only bool indicating if the drawing tool is attached to an indicator or strategy |
| IgnoresUserInput | A read-only bool determining if the drawing tool can be interacted with by the user. |
| IsGlobalDrawingTool | A bool determining if the drawing tool displays on all charts of the instrument |
| IsLocked | A bool determining if the drawing tool can be moved |
| IsSeparateZOrder | A bool determining if the drawing tool will reside on a different ZOrder from the NinjaScript object it was drawn |
| IsUserDrawn | A read-only bool indicating if drawing tool was manually drawn by a user |
| PanelIndex | An int value representing the panel the drawing tool resides |
| SupportsAlerts | A read-only bool indicating if the drawing tool can be used for creating an alert |
| Tag | A string value representing the unique ID of the draw object. (Global draw objects will have an "@" added as a prefix to the string) |
| ZOrderType | A read-only enum indicating the order in which the drawing tool will be drawn.    Possible values are:  •DrawingToolZOrder.Normal,  •DrawingToolZOrder.AlwaysDrawnFirst,  •DrawingToolZOrder.AlwaysDrawnLast |

**ChartAnchor Properties**

|  |  |
| --- | --- |
| <ChartAnchor>.BarsAgo | An int representing the "barsAgo" value that was passed to the Draw method    **Note**:  This value will **NOT** be set for objects drawn manually |
| <ChartAnchor>.DisplayName | A string representing name of the DrawingTool's chart anchor that is displaying on the UI |
| <ChartAnchor>.DrawingTool | The IDrawingTool object which created the DrawingTool's chart anchor object |
| <ChartAnchor>.DrawnOnBar | An int representing the [CurrentBar](https://ninjatrader.com/es/support/helpGuides/nt8/currentbar.htm) value that the DrawingTool's chart anchor was drawn |
| <ChartAnchor>.IsNinjaScriptDrawn | A bool indicating the object was drawn programmatically |
| <ChartAnchor>.Price | A double representing the price the  DrawingTool's chart anchor was drawn |
| <ChartAnchor>.SlotIndex | A double representing the DrawingTool's chart anchor index value the anchor was drawn |
| <ChartAnchor>.Time | A DateTime representing the time value the DrawingTool's chart anchor was drawn |

**Examples**

| ns | |
| --- | --- |
| Text myText; protected override void OnBarUpdate() {     if(CurrentBar == 50)     myText = Draw.Text(this, "tag", "test", 0, High[0]);         if(myText != null)   {               Print(myText.Anchor.DrawnOnBar); // drawn on bar 50   }   } | |
| **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) >  **IDrawingTool** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawingtools_drawobjects.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/pricelevels.htm) |

**Definition**

Represents an interface that exposes information regarding a drawn chart object.

IDrawingTool Properties are standard properties that are shared by all drawing tools.

Each specific **IDrawingTool** will have its own uniquely named **ChartAnchor**representing where the object was drawn on the chart.  The name and number of **ChartAnchors** will be specific to that drawing tool (e.g., StartAnchor, EndAnchor, etc), however the fields available will be the same (e.g., BarsAgo, DrawnOnBar, etc).  Details on those shared fields are outlined in the **ChartAnchor Properties** section toward the bottom of this topic.

|  |
| --- |
| **Note**:  For implementing a custom Drawing Tool project, please see the [DrawingTools](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) section of this help guide. |

**IDrawingTool Properties**

|  |  |
| --- | --- |
| Anchors | A read-only collection of all of the [IDrawingTool's ChartAnchors](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#chartanchor) |
| AttachedTo | An enum determining where the drawing tool is attached.    Possible values are:  •AttachedToType.Bars,  •AttachedToType.GlobalInstrument,  •AttachedToType.Indicator,  •AttachedToType.Strategy |
| DrawingState | The current [DrawingState](https://ninjatrader.com/es/support/helpGuides/nt8/drawingstate.htm) of the drawing tool |
| DrawnBy | An object value indicating which type of NinjaScript the drawing tool originated (null if user drawn) |
| IsAttachedToNinjaScript | A read-only bool indicating if the drawing tool is attached to an indicator or strategy |
| IgnoresUserInput | A read-only bool determining if the drawing tool can be interacted with by the user. |
| IsGlobalDrawingTool | A bool determining if the drawing tool displays on all charts of the instrument |
| IsLocked | A bool determining if the drawing tool can be moved |
| IsSeparateZOrder | A bool determining if the drawing tool will reside on a different ZOrder from the NinjaScript object it was drawn |
| IsUserDrawn | A read-only bool indicating if drawing tool was manually drawn by a user |
| PanelIndex | An int value representing the panel the drawing tool resides |
| SupportsAlerts | A read-only bool indicating if the drawing tool can be used for creating an alert |
| Tag | A string value representing the unique ID of the draw object. (Global draw objects will have an "@" added as a prefix to the string) |
| ZOrderType | A read-only enum indicating the order in which the drawing tool will be drawn.    Possible values are:  •DrawingToolZOrder.Normal,  •DrawingToolZOrder.AlwaysDrawnFirst,  •DrawingToolZOrder.AlwaysDrawnLast |

**ChartAnchor Properties**

|  |  |
| --- | --- |
| <ChartAnchor>.BarsAgo | An int representing the "barsAgo" value that was passed to the Draw method    **Note**:  This value will **NOT** be set for objects drawn manually |
| <ChartAnchor>.DisplayName | A string representing name of the DrawingTool's chart anchor that is displaying on the UI |
| <ChartAnchor>.DrawingTool | The IDrawingTool object which created the DrawingTool's chart anchor object |
| <ChartAnchor>.DrawnOnBar | An int representing the [CurrentBar](https://ninjatrader.com/es/support/helpGuides/nt8/currentbar.htm) value that the DrawingTool's chart anchor was drawn |
| <ChartAnchor>.IsNinjaScriptDrawn | A bool indicating the object was drawn programmatically |
| <ChartAnchor>.Price | A double representing the price the  DrawingTool's chart anchor was drawn |
| <ChartAnchor>.SlotIndex | A double representing the DrawingTool's chart anchor index value the anchor was drawn |
| <ChartAnchor>.Time | A DateTime representing the time value the DrawingTool's chart anchor was drawn |

**Examples**

| ns | |
| --- | --- |
| Text myText; protected override void OnBarUpdate() {     if(CurrentBar == 50)     myText = Draw.Text(this, "tag", "test", 0, High[0]);         if(myText != null)   {               Print(myText.Anchor.DrawnOnBar); // drawn on bar 50   }   } | |
| **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) >  **Stroke Class** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/scalejustification.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chart.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/usercontrolcollection.htm) |

**Definition**

Objects derived from the Stroke class are used to characterize how a plot is visually displayed (plotted) on a chart.

**Syntax**

Stroke(Stroke *stroke*)

Stroke(Brush *brush*)

Stroke(Brush *brush*, float *width*)

Stroke(Brush *brush*, DashStyle *dashStyleHelper*, float *width*)

**Parameters**

|  |  |
| --- | --- |
| brush | The brush used to draw the plot ([reference](http://msdn.microsoft.com/en-us/library/System.Windows.Media.Brush%28v=vs.110%29.aspx)) |
| dashStyleHelper | Possible values:    DashStyleHelper.Dash  DashStyleHelper.DashDot  DashStyleHelper.DashDotDot  DashStyleHelper.Dot  DashStyleHelper.Solid |
| stroke | The [stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object |
| width | The width of the stroke |

**Properties**

|  |  |
| --- | --- |
| Brush | The System.Windows.Media.Brush used to construct the stroke ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx)) |
| BrushDX | A [SharpDX.Direct2D1.Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) used to actually render the stroke    **Note**:  To avoid and resolve access violation exceptions, please see Warning and examples remarked below |
| DashStyleDX | A [SharpDX.Direct2D1.DashStyle](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_dashstyle.htm) u sed to render the stroke style    **Note**:  To avoid and resolve access violation exceptions, please see Warning and examples remarked below |
| DashStyleHelper | A dashstyle used to construct the stroke. Possible values are:    •DashStyleHelper.Dash  •DashStyleHelper.DashDot  •DashStyleHelper.DashDotDot  •DashStyleHelper.Dot  •DashStyleHelper.Solid |
| RenderTarget | The [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/rendertarget.htm) drawing context used for the stroke.    **Note**: This property must be set before accessing a stroke's BrushDX property. Please see Warning and examples remarked below |
| StrokeStyle | A [SharpDX.Direct2D1.StrokeStyle](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle.htm) |
| Width | A float |

|  |
| --- |
| **Warning**:  There may be situations where a **RenderTarget** has not been set, and to prevent access violation exception before accessing the **BrushDX** or **DashStyleDX** properties, you should explicitly set the **RenderTarget** before attempting to access that property.  Please see the example below. |

**Examples**

See the [AddPlot()](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm) method for additional examples.

| ns **Using a Stroke SharpDX Brush for Custom Rendering** |
| --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {     IsOverlay = true;     // set the Stroke default to red brush     MyStroke = new Stroke(Brushes.Red);   }   else if (State == State.Configure)   {   } }   public override void OnRenderTargetChanged() {   // Explicitly set the Stroke RenderTarget   if (RenderTarget != null)     MyStroke.RenderTarget = RenderTarget; }   protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // create two points from the top left corner   SharpDX.Vector2 pointA = new SharpDX.Vector2(0, 0);   // to 300 pixels offset X and Y to create a diagonal line   SharpDX.Vector2 pointB = new SharpDX.Vector2(300, 300);     // Draw the line using the Stroke SharpDX brush   RenderTarget.DrawLine(pointA, pointB, MyStroke.BrushDX, MyStroke.Width, MyStroke.StrokeStyle);   }   [NinjaScriptProperty] [Description("My Stroke")] public Stroke MyStroke { get; set; } |

| ns **Convert the Windows Media Brush to a SharpDX Brush** | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {     IsOverlay = true;     // set stroke default to blue brush     MyStroke = new Stroke(Brushes.Blue);   }   else if (State == State.Configure)   {   } }   protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // create two points from the top left corner   SharpDX.Vector2 pointA = new SharpDX.Vector2(0, 0);   // to 300 pixels offset X and Y to create a diagonal line   SharpDX.Vector2 pointB = new SharpDX.Vector2(300, 300);     NinjaTrader.Gui.Stroke MyStroke = new Stroke(Brushes.Blue);     // if BrushDX is null, convert the constructed brush to a DX brush   SharpDX.Direct2D1.Brush myBrush = MyStroke.BrushDX ?? MyStroke.Brush.ToDxBrush(RenderTarget);   RenderTarget.DrawLine(pointA, pointB, myBrush, MyStroke.Width, MyStroke.StrokeStyle);     myBrush.Dispose(); }   [NinjaScriptProperty] [Description("My Stroke")] public Stroke MyStroke { get; set; } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) >  **Code Breaking Changes** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript_best_practices.htm) |

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

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

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

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

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

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

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

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

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

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

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

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

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

**Code Breaking Table**

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

Show 255075100All entries

Filter results:

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

Showing 1 to 25 of 197 entries

Previous12345…8Next

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