|  |  |
| --- | --- |
| **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) > [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) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.Arc()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/andrewspitchfork.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/arc.htm) |

**Definition**

Draws an arc.

**Method Return Value**

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

**Syntax**

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

**Parameters**

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

**Examples**

| ns | |
| --- | --- |
| // Draws a dotted lime green arc Draw.Arc(this, "tag1", false, 10, 1000, 0, 1001, Brushes.LimeGreen, DashStyleHelper.Dot, 2); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.ExtendedLine()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ellipse.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/extendedline.htm) |

**Definition**

Draws a line with infinite end points.

**Method Return Value**

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

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

**Parameters**

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

**Examples**

| ns | |
| --- | --- |
| // Draws a dotted lime green Draw.ExtendedLine(this, "tag1", 10, Close[10], 0, Close[0], Brushes.LimeGreen, DashStyleHelper.Dot, 2); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.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) > [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.ArrowLine()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/arrowdown.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/arrowline.htm) |

**Definition**

Draws an arrow line.

**Method Return Value**

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

**Syntax**

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

**Parameters**

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

**Examples**

| ns | |
| --- | --- |
| // Draws a dotted lime green arrow line Draw.ArrowLine(this, "tag1", 10, 1000, 0, 1001, Brushes.LimeGreen, DashStyleHelper.Dot, 2); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) >  **Draw.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.HorizontalLine()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/gannfan.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/horizontalline.htm) |

**Definition**

Draws a horizontal line.

**Method Return Value**

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

**Syntax**

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

**Parameters**

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

**Examples**

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

**Definition**

Draws a vertical line.

**Method Return Value**

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

**Syntax**

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

**Parameters**

|  |  |
| --- | --- |
| owner | The hosting NinjaScript object which is calling the draw method    Typically will be the object which is calling the draw method (e.g., "this") |
| tag | A user defined unique id used to reference the draw object.    For example, if you pass in a value of "myTag", each time this tag is used, the same draw object is modified. If unique tags are used each time, a new draw object will be created each time. |
| barsAgo | The bar the object will be drawn at. A value of 10 would be 10 bars ago. |
| time | The time the object will be drawn at. |
| brush | The brush used to color draw object ([reference](https://msdn.microsoft.com/en-us/library/system.windows.media.brushes%28v=vs.110%29.aspx)) |
| 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) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) > [StrokeStyle](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle.htm) >  **SharpDX.Direct2D1.StrokeStyle.DashStyle** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_dashoffset.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_endcap.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | | |

**Definition**

Gets a value that describes the stroke's dash pattern.

(See also [unmanaged API documentation](http://msdn.microsoft.com/en-us/library/dd372217.aspx))

|  |
| --- |
| **Note:**If a custom dash style is specified, the dash pattern is described by the dashes array, which can be retrieved by calling the [GetDashes()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_getdashes.htm) method. |

**Property Value**

A SharpDX.Direct2D1.DashStyle enum value that describes the predefined dash pattern used, or DashStyle.Custom if a custom dash style is used.

Possible Values are:

|  |  |
| --- | --- |
| Solid | A solid line with no breaks. |
| Dash | A dash followed by a gap of equal length. The dash and the gap are each twice as long as the stroke thickness.  The equivalent dash array for Dash is {2, 2}. |
| Dot | A dot followed by a longer gap.  The equivalent dash array for Dot is {0, 2}. |
| DashDot | A dash, followed by a gap, followed by a dot, followed by another gap. The equivalent dash array for DashDot is {2, 2, 0, 2}. |
| DashDotDot | A dash, followed by a gap, followed by a dot, followed by another gap, followed by another dot, followed by another gap. The equivalent dash array for DashDotDot is {2, 2, 0, 2, 0, 2}. |
| Custom | The dash pattern is specified by an array of floating-point values. |

(See also [unmanaged API documentation](http://msdn.microsoft.com/en-us/library/dd368087.aspx))

**Syntax**

<StrokeStyle>.DashStyle

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

**Definition**

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

**Method Return Value**

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

**Syntax**

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

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

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

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

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

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

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

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

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

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

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

Draw.PathTool(NinjaScriptBase owner, string tag, bool isAutoScale, List<ChartAnchor> chartAnchors, 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. |
| 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. |
| 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 |
| 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 |
| 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)) |
| 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 PathTool object based on bars ago and y anchors Draw.PathTool(this, "tag1", false, 20, 194, 10, 184, 13, 176, 25, 182);    // Draws a PathTool 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.PathTool(this, "tag1", false, anchors, Brushes.CornflowerBlue, DashStyleHelper.Solid, 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.PathTool()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/line.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/pathtool.htm) |

**Definition**

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

**Method Return Value**

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

**Syntax**

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

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

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

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

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

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

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

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

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

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

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

Draw.PathTool(NinjaScriptBase owner, string tag, bool isAutoScale, List<ChartAnchor> chartAnchors, 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. |
| 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. |
| 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 |
| 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 |
| 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)) |
| 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 PathTool object based on bars ago and y anchors Draw.PathTool(this, "tag1", false, 20, 194, 10, 184, 13, 176, 25, 182);    // Draws a PathTool 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.PathTool(this, "tag1", false, anchors, Brushes.CornflowerBlue, DashStyleHelper.Solid, 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.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:**  »No topics above this level«  **Entering Calculation Logic** | | [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout.htm) |

The [OnBarUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onbarupdate.htm) method is called for each incoming tick, or on the close of a bar (if enabled) when performing real-time calculations, and is called on each bar of a [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) object when re-calculating the indicator (For example, an indicator would be re-calculated when adding it to an existing chart that has existing price data displayed). This is the main method called for indicator calculation, and we will calculate the CCI value and set the conditions used to draw the CCI plot within this method.

The [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method is called once before any bar data is loaded, and is used to configure the indicator (among other things).

**Initializing the Indicator**  
The code below is automatically generated by the wizard and added to the OnStateChange() method, within State.SetDefaults. It configures the indicator for one plot and five lines, and sets the parameters entered in the wizard:

| ns |
| --- |
| AddLine(Brushes.DimGray, 200, "Level 2"); AddLine(Brushes.DimGray, 100, "Level 1"); AddLine(Brushes.DimGray, 0, "Zero Line"); AddLine(Brushes.DimGray, -100, "Level -1"); AddLine(Brushes.DimGray, -200, "Level -2"); |

To change the visual properties of the Zero Line, replace the third line in the code above with the line below. This will change the color to black and the line style to "dash:"

| ns |
| --- |
| AddLine(new Stroke(Brushes.Black, DashStyleHelper.Dash,2), 0, "Zero Line"); |

The code above uses an alternative method overload (an alternative set of arguments passed in to the AddLine() method), in order to pass in a [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object rather than a [Brush](http://sharpdx.org/documentation/api/t-sharpdx-direct2d1-brush). With a Stroke, not only can we still specify a Brush, but we have additional options to change the dash style (via DashStyleHelper) and the line width. After this change, your configured lines and plots should look like this:

| ns |
| --- |
| AddLine(Brushes.DimGray, 200, "Level 2"); AddLine(Brushes.DimGray, 100, "Level 1"); AddLine(new Stroke(Brushes.Black, DashStyleHelper.Dash,2), 0, "Zero Line"); AddLine(Brushes.DimGray, -100, "Level -1"); AddLine(Brushes.DimGray, -200, "Level -2"); |

**Adding Core Indicator Logic**

Since this tutorial is meant to cover custom drawing and manually changing properties within an indicator, we will not go too in-depth into the core calculation logic for this custom CCI. Instead, we will copy and paste the core calculation logic from the @CCI indicator already built-in to NinjaTrader.

The @CCI indicator uses an SMA object in its calculations. To add this, copy the line below from @CCI into your custom CCI, directly below the class declaration:

| ns |
| --- |
| private SMA sma; |

Next, copy the following initialization for the SMA object into the OnStateChange() method, within State.Configure:

| ns |
| --- |
| sma = SMA(Typical, Period); |

Next, copy the core calculation logic from @CCI into the OnBarUpdate() method of your custom indicator:

| ns |
| --- |
| if (CurrentBar == 0)   Value[0] = 0; else {   double mean = 0;   double sma0 = sma[0];     for (int idx = Math.Min(CurrentBar, Period - 1); idx >= 0; idx--)       mean += Math.Abs(Typical[idx] - sma0);     Value[0] = (Typical[0] - sma0) / (mean.ApproxCompare(0) == 0 ? 1 : (0.015 \* (mean / Math.Min(Period, CurrentBar + 1)))); } |

The code for your MyCCI class should now look as follows (in addition to the using statements and class declaration) :

| ns |
| --- |
| public class MyCCI : Indicator {     private SMA sma;     protected override void OnStateChange()   {       if (State == State.SetDefaults)       {           Description                 = @"NinjaScript Custom Drawing Indicator Tutorial";           Name                       = "MyCCI";           Calculate                   = Calculate.OnBarClose;           IsOverlay                   = false;           DisplayInDataBox           = true;           DrawOnPricePanel           = true;           DrawHorizontalGridLines     = true;           DrawVerticalGridLines       = true;           PaintPriceMarkers           = true;           ScaleJustification         = NinjaTrader.Gui.Chart.ScaleJustification.Right;           //Disable this property if your indicator requires custom values that cumulate with each new market data event.           //See Help Guide for additional information.           IsSuspendedWhileInactive   = true;           Period                 = 14;           AddPlot(Brushes.Orange, "CCI");           AddLine(Brushes.DimGray, 200, "Level 2");           AddLine(Brushes.DimGray, 100, "Level 1");           AddLine(new Stroke(Brushes.Black, DashStyleHelper.Dash,2), 0, "Zero Line");           AddLine(Brushes.DimGray, -100, "Level -1");           AddLine(Brushes.DimGray, -200, "Level -2");         }       else if (State == State.Configure)       {           sma = SMA(Typical, Period);       }   }     protected override void OnBarUpdate()   {       if (CurrentBar == 0)           Value[0] = 0;       else       {           double mean = 0;           double sma0 = sma[0];             for (int idx = Math.Min(CurrentBar, Period - 1); idx >= 0; idx--)               mean += Math.Abs(Typical[idx] - sma0);             Value[0] = (Typical[0] - sma0) / (mean.ApproxCompare(0) == 0 ? 1 : (0.015 \* (mean / Math.Min(Period, CurrentBar + 1))));       }   } |

**Custom Drawing**  
Add the following code into the OnBarUpdate() method, directly beneath the core calculation logic:

| ns |
| --- |
| // if the plot value is greater than 100, paint the plot green at that bar index if (Value[0] > 100)   PlotBrushes[0][0] = Brushes.Green;   // if the plot value is less than -100, paint the plot red at that bar index if (Value[0] < -100)   PlotBrushes[0][0] = Brushes.Red;   // if the plot value is between 100 and -100, paint the plot orange at that bar index if (Value[0] >= -100 && Value[0] <= 100)   PlotBrushes[0][0] = Brushes.Orange; |

This will conditionally change the color of the CCI plot (referenced by Values[0]) based on its value. By using PlotBrushes[0][0], we are specifying that we wish to change the color of the first plot in the collection at a specific bar index (the current bar index each time the condition is triggered), and we wish for the plot the remain that color at that index, even if the plot value changes in the future. If instead we wished to change the entire plot color, we could use Plots[0].Brush.

PlotBrushes holds a collection of brushes used for the various plots in the indicator. In addition to this, there are several other collections that serve similar purposes, which can be used in the same way. Some examples of these collections are below:

|  |  |
| --- | --- |
| [BackBrushes](https://ninjatrader.com/es/support/helpGuides/nt8/backbrushes.htm) | A collection of Brushes used for chart background color at specific bar indexes |
| [BarBrushes](https://ninjatrader.com/es/support/helpGuides/nt8/barbrushes.htm) | A collection of Brushes used to paint bars at specific indexes |
| [CandleOutlineBrushes](https://ninjatrader.com/es/support/helpGuides/nt8/candleoutlinebrushes.htm) | A collection of Brushes used to paint candle outlines at specific indexes |

Now that everything is in place, your class code should look as below. You are now ready to [compile the indicator](https://ninjatrader.com/es/support/helpGuides/nt8/compiling6.htm) and configure it on a chart.

| ns | |
| --- | --- |
| public class MyCCI : Indicator {     private SMA sma;     protected override void OnStateChange()   {       if (State == State.SetDefaults)       {           Description                 = @"NinjaScript Custom Drawing Indicator Tutorial";           Name                       = "MyCCI";           Calculate                   = Calculate.OnBarClose;           IsOverlay                   = false;           DisplayInDataBox           = true;           DrawOnPricePanel           = true;           DrawHorizontalGridLines     = true;           DrawVerticalGridLines       = true;           PaintPriceMarkers           = true;           ScaleJustification         = NinjaTrader.Gui.Chart.ScaleJustification.Right;           //Disable this property if your indicator requires custom values that cumulate with each new market data event.           //See Help Guide for additional information.           IsSuspendedWhileInactive   = true;           Period                 = 14;           AddPlot(Brushes.Orange, "CCI");           AddLine(Brushes.DimGray, 200, "Level 2");           AddLine(Brushes.DimGray, 100, "Level 1");           AddLine(new Stroke(Brushes.Black, DashStyleHelper.Dash,2), 0, "Zero Line");           AddLine(Brushes.DimGray, -100, "Level -1");           AddLine(Brushes.DimGray, -200, "Level -2");         }       else if (State == State.Configure)       {           sma = SMA(Typical, Period);       }   }     protected override void OnBarUpdate()   {       if (CurrentBar == 0)           Value[0] = 0;       else       {           double mean = 0;           double sma0 = sma[0];             for (int idx = Math.Min(CurrentBar, Period - 1); idx >= 0; idx--)               mean += Math.Abs(Typical[idx] - sma0);             Value[0] = (Typical[0] - sma0) / (mean.ApproxCompare(0) == 0 ? 1 : (0.015 \* (mean / Math.Min(Period, CurrentBar + 1))));       }         if (Value[0] > 100)           PlotBrushes[0][0] = Brushes.Green;         if (Value[0] < -100)           PlotBrushes[0][0] = Brushes.Red;         if (Value[0] >= -100 && Value[0] <= 100)           PlotBrushes[0][0] = Brushes.Orange;   } | |
| **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) > [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) > [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) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) >  **SharpDX.Direct2D1.StrokeStyleProperties** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_startcap.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_sweepdirection.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | |

**Definition**

Describes the stroke that outlines a shape.

(See also [unmanaged API documentation](http://msdn.microsoft.com/en-us/library/dd368164.aspx))

**Syntax**

struct StrokeStyleProperties

**Properties**

|  |  |
| --- | --- |
| StartCap | The [StartCap](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_startcap.htm) value applied to the start of all the open figures in a stroked geometry. |
| EndCap | The [EndCap](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_endcap.htm) value applied to the end of all the open figures in a stroked geometry. |
| DashCap | The [DashCap](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_dashcap.htm) value for the shape at either end of each dash segment. |
| LineJoin | A [LineJoin](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_linejoin.htm) value that describes how segments are joined. This value is ignored for a vertex if the segment flags specify that the segment should have a smooth join. |
| MiterLimit | The [MeterLImit](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_miterlimit.htm) value of the thickness of the join on a mitered corner. This value is always treated as though it is greater than or equal to 1.0f. |
| DashStyle | A [DashStyle](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_dashstyle.htm) value that specifies whether the stroke has a dash pattern and, if so, the dash style. |
| DashOffset | A [DashOffset](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_dashoffset.htm) value that specifies an offset in the dash sequence. A positive dash offset value shifts the dash pattern, in units of stroke width, toward the start of the stroked geometry. A negative dash offset value shifts the dash pattern, in units of stroke width, toward the end of the stroked geometry. |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) > [Draw.ArrowLine()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_arrowline.htm) >  **ArrowLine** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_arrowline.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_arrowline.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_arrowup.htm) |

**Definition**

Represents an interface that exposes information regarding an Arrow 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 | |
| --- | --- |
| // Draw an ArrowLine object ArrowLine myArrow = Draw.ArrowLine(this, "myArrowLine", 3, High[3], 1, High[1], Brushes.Blue, DashStyleHelper.DashDot, 3);   // Disable the arrow's visibility myArrow.IsVisible = false; | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) > [Draw.ExtendedLine()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_extendedline.htm) >  **ExtendedLine** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_extendedline.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_extendedline.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_fibonaccicircle.htm) |

**Definition**

Represents an interface that exposes information regarding an Extended 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 dotted lime green Extended Line ExtendedLine myLine = Draw.ExtendedLine(this, "tag1", 10, Close[10], 0, Close[0], Brushes.LimeGreen, DashStyleHelper.Dot, 2);   // Make the line a Global Drawing Object myLine.IsGlobalDrawingTool = 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.HorizontalLine()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_horizontalline.htm) >  **HorizontalLine** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_horizontalline.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_horizontalline.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_line.htm) |

**Definition**

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

**Methods and Properties**

|  |  |
| --- | --- |
| StartAnchor | An [IDrawingTool's ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/idrawingtool.htm#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 HorizontalLine object HorizontalLine myLine = Draw.HorizontalLine(this, "tag1", 1000, Brushes.Black);   // Set a new Stroke for the object myLine.Stroke = new Stroke(Brushes.Green, DashStyleHelper.Dash, 5); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) > [Draw.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) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) >  **SharpDX.Direct2D1.StrokeStyle** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_solidcolorbrush_color.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_dashcap.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | | |

**Definition**

Describes the caps, miter limit, line join, and dash information for a stroke.

(See also [unmanaged API documentation](http://msdn.microsoft.com/en-us/library/dd372217.aspx))

|  |
| --- |
| **Notes**:  1.A stroke style is a device-independent resource; you can create it once then retain it for the life of your application. Please see the [MSDN Direct2D Resources Overview](https://msdn.microsoft.com/en-us/library/dd756757(v=vs.85).aspx) for more information.  2.For convenience, **Direct2D** provides the [StrokeStyleProperties](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyleproperties.htm) function for creating new a **StrokeStyle**. |

**Syntax**

class StrokeStyle

**Constructors**

|  |  |
| --- | --- |
| new StrokeStyle(Factory factory, [StrokeStyleProperties](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyleproperties.htm) properties) | Creates an **StrokeStyle** that describes start cap, dash pattern, and other features of a stroke. |
| new StrokeStyle(Factory factory, [StrokeStyleProperties](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyleproperties.htm) properties, float[] dashes) | Creates an **StrokeStyle** that describes start cap, dash pattern, and other features of a stroke. |

|  |
| --- |
| **Tip**: For NinjaScript development purposes, when creating a **StrokeStyle**objectyou should use the [NinjaTrader.Core.Globals.D2DFactory](https://ninjatrader.com/es/support/helpGuides/nt8/d2dfactory.htm) property |

**Method and Properties**

|  |  |
| --- | --- |
| [DashCap](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_dashcap.htm) | Gets a value that specifies how the ends of each dash are drawn. |
| [DashesCount](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_dashescount.htm) | Retrieves the number of entries in the dashes array. |
| [DashOffset](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_dashoffset.htm) | Retrieves a value that specifies how far in the dash sequence the stroke will start. |
| [DashStyle](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_dashstyle.htm) | Gets a value that describes the stroke's dash pattern. |
| [Dispose()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase_dispose.htm) | Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from [SharpDX.DisposeBase](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase.htm).) |
| [EndCap](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_endcap.htm) | Retrieves the type of shape used at the end of a stroke. |
| [GetDashes()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_getdashes.htm) | Copies the dash pattern to the specified array. |
| [IsDisposed](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase_isdisposed.htm) | Gets a value indicating whether this instance is disposed. (Inherited from DisposeBase.) |
| [LineJoin](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_linejoin.htm) | Retrieves the type of joint used at the vertices of a shape's outline. |
| [MiterLimit](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_miterlimit.htm) | Retrieves the limit on the ratio of the miter length to half the stroke's thickness. |
| [StartCap](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_startcap.htm) | Retrieves the type of shape used at the beginning of a stroke. |
| **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) >  **Code Breaking Changes** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript_best_practices.htm) |

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

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

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

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

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

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

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

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

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

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

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

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

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

**Code Breaking Table**

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

Show 255075100All entries

Filter results:

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

Showing 1 to 25 of 197 entries

Previous12345…8Next

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

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [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) >  **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) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **DrawingState** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/dispose.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawnby.htm) |

**Definition**

Represents the current state of the drawing tool to perform various actions, such as building, editing, or moving.

**Property Values**

An enum representing the current state of the drawing tool.  Possible values are:

|  |  |
| --- | --- |
| DrawingState.Building | The initial state when a drawing tool is first being drawn, allowing for the anchors to be set for the drawing. |
| DrawingState.Editing | Allows for changing the values of any of the drawing tools anchors |
| DrawingState.Normal | The drawing tool is normal on the chart and is not in a state to allow for changes. |
| DrawingState.Moving | The entire drawing tool to be moved by a user. |

**Syntax**

DrawingState

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseDown(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, Point point) {   switch(DrawingState)   {                             case DrawingState.Normal:         DrawingState = DrawingState.Editing; // set state to allow editing         break;             case DrawingState.Editing:         // do your edits here         break;     case DrawingState.Moving:         return; // don't allow move whe editing               }       } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Distribution](https://ninjatrader.com/es/support/helpGuides/nt8/distribution.htm) >  **Considerations For Compiled Assemblies** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/distribution.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/distribution.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/import.htm) |

**Using Compiled Assemblies**

Compiled assemblies (DLL's) allow you to bundle your scripts into a format that hides your proprietary code along with any supporting resources. Compiled assemblies provide distinct benefits, especially for commercially distributed code, but there are a few considerations to keep in mind. Typecasting and building resource files (sounds, images, etc.) into your assemblies must be approached differently to ensure cleanly packaged, error-free DLL's.

**Casting Types in a DLL (Using dynamic Types)**

Sometimes, you may need to cast your objects to NinjaScript types, such as when iterating through the DrawObjects collection to obtain a reference to a particular Drawing Object on a chart. When running C# code which has not been compiled into an assembly, typecasting can be done normally, as in the example below:

| ns **Typecasting in code outside of a compiled assembly** |
| --- |
| **protected** **override** **void** OnBarUpdate() {   **foreach**(HorizontalLine line **in** DrawObjects)   {       *// Print the tag of each Horizontal Line on the chart*       Print(String.Format("Horizontal Line {0} found.", line.Tag));   } } |

An obstacle arises with traditional typecasting in a compiled assembly, since the NinjaScript Type you attempt to cast will be present in both your DLL and NinjaTrader's Custom.dll assembly. If you plan to compile your code into a DLL, you will need to use the [dynamic Type](https://msdn.microsoft.com/en-us/library/dd264741.aspx) to avoid this conflict by dynamically assigning the Type at runtime, using the guidelines below:

1.Define a variable of Type 'dynamic'

2.Assign a reference to the needed object to the dynamic variable

3.Access the dynamic variable as if it were of the expected Type

| ns D**ynamic variables as an alternative to typecasting inside of a compiled assembly** |
| --- |
| foreach (dynamic line in DrawObjects.ToList()) {   // Use ToString().Equals() to detect the object's Type   if (line.ToString().Equals("NinjaTrader.NinjaScript.DrawingTools.HorizontalLine"))   {       // Access the object by assuming that it is the Type we expect       Print(String.Format("Horizontal Line {0} detected!", line.Tag));   } } |

The above dynamic approach will work for primitive types. For instantiating more complex types / classes though, such as adding a new [PriceLevel](https://ninjatrader.com/es/support/helpGuides/nt8/pricelevels.htm) programmatically to an existing drawing tool, [Reflection](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/reflection) would need to used.

| ns **Instantiating more complex types such as the PriceLevels class inside of a compiled assembly** |
| --- |
| foreach (dynamic dt in DrawObjects.ToList()) {   if(dt.ToString().Equals("NinjaTrader.NinjaScript.DrawingTools.FibonacciRetracements"))   {     Type type         = dt.PriceLevels.GetType().GetGenericArguments()[0];     Assembly assembly = type.Assembly;     var pl           = assembly.CreateInstance(type.FullName, false, BindingFlags.CreateInstance, null, new object[] { 55.5, Brushes.Red, 2 }, new       System.Globalization.CultureInfo("en-US"), new object[] {});     dt.PriceLevels.GetType().GetMethod("Add").Invoke(dt.PriceLevels, new object[] { pl } );     this.ForceRefresh();   } } |

**Working With the dynamic Type**

Using dynamic variables in the technique above requires careful attention to accessing members appropriately, and thus should be avoided if you do not intend to use or distribute compiled assemblies.

•**No Intelliprompt**: Since the compiler cannot know which Type you assume a dynamic variable to be, no intelliprompt will be displayed to help search through Type members. The same applies to Visual Studio's Intellisense or similar utilities

•**No Compile Errors**: For the same reason, the compiler cannot know if you are using the variable in a way not supported by its expected Type, trying to access members not present in that Type, or other related errors. Thus, any such errors which would be caught by the compiler when typecasting will be missed, and will result in runtime errors instead. If a runtime error were to be triggered, the error may be more difficult to interpret.

oExample: If you tried to access "line.tag" (improper capitalization) in the examples above, you would receive the following errors:

▪Typecasting / Compile Error: *"'NinjaTrader.NinjaScript.DrawingTools.HorizontalLine' does not contain a definition for 'tag' and no extension method accepting a first argument of type 'NinjaTrader.NinjaScript.DrawingTools.HorizontalLine' can be found (are you missing a using directive or an assembly reference?)"*

▪dynamic / Runtime Error: *"Error on calling 'OnBarUpdate' method on bar 0: 'NinjaTrader.NinjaScript.DrawingTools.DrawingTool.tag' is inaccessible due to its protection level"*

**Adding XAML and Other Files Into a DLL**

When [exporting a compiled assembly](https://ninjatrader.com/es/support/helpGuides/nt8/export.htm) through NinjaTrader, no additional resource files can be added. There are two ways around this. The first is to export the DLL from NinjaTrader, then open the exported .zip file, add any additional files, and re-zip the archive, but this will result in your resource files being fully accessible to end users. The second and recommended approach is to use a fully featured IDE such as Visual Studio to build your DLL's.

For more information on how to accomplish this with Visual Studio, see the "AddOn Development Environment" section of the [AddOn Development Overview](https://ninjatrader.com/es/support/helpGuides/nt8/addon_development_overview.htm) page. Although the page focuses on AddOn development, the sample project it provides can be used to develop other NinjaScript Types, as well.

**Exporting custom drawing tools as assembly / DLL**

When planning to distribute your custom drawing tools via assemblies, please understand it's paramount that you implement your own Draw. method to allow the drawing tool getting called programmatically by other NinjaScript objects.

The NinjaTrader default drawing tools would implement this via a partial class, for example you would see -

| ns **Default**NinjaTrader **drawing tool Draw. method handling** |
| --- |
| public static partial class Draw {  } |

However since partial classes could not span across two assemblies, therefore a custom non partial Draw. method for your NinjaScript drawing tool would be needed.

| ns **Custom drawing tool Draw. method handling** |
| --- |
| public static class MyDrawCustom {  } |

**Exports might not be backwards compatible**

NinjaScript exports might not be backwards compatible with previous versions of NinjaTrader.

This is known to happen every time a new type (e.g. Enum) was introduced, since the newly introduced types are not known to prior releases of NinjaTrader

Typically an error message like the following would be seen:

*"Error on calling 'SetState' method: Could not load type 'NinjaTrader.NinjaScript.Indicators.CumulativeDeltaType' from assembly 'NinjaTrader.Vendor, Version=8.0.12.0, Culture=neutral, PublicKeyToken=null'."*

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **DrawingTool** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/displayname.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawnonbar.htm) |

**Definition**

The DrawingTool object which owns a chart anchor.

**Property Value**

A IDrawingTool object representing the owner of the chart anchor

**Syntax**

<ChartAnchor>.DrawingTool

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)      {  Name = "SampleDrawingTool";  MyAnchor = new ChartAnchor();  MyAnchor.DrawingTool = this; //NinjaTrader.NinjaScript.DrawingTools.SampleDrawingTool      }      else if (State == State.Configure)      {        } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **ChartAnchor** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/attachedto.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/copydatavalues.htm) |

**Definition**

Defines objects used by Drawing Tools which represent a point on the chart where the Drawing Tool is located.

**Syntax**

class ChartAnchor

**Constructors**

|  |  |
| --- | --- |
| new ChartAnchor() | Initializes a new instance of the ChartAnchor object |
| new ChartAnchor(DateTime time, double price, [ChartControl](https://ninjatrader.com/es/support/helpGuides/nt8/chartcontrol.htm) chartControl) | Initializes a new instance of the ChartAnchor object using time, price, and relative chart control |
| new ChartAnchor(DateTime time, double yValue, int currentBar, [ChartControl](https://ninjatrader.com/es/support/helpGuides/nt8/chartcontrol.htm) chartControl) | Initializes a new instance of the ChartAnchor object using time, y-axis coordinates, current bar, and relative chart control |

**Methods and Properties**

|  |  |
| --- | --- |
| [CopyDataValues()](https://ninjatrader.com/es/support/helpGuides/nt8/copydatavalues.htm) | Copies the ChartAnchor time and price values from on anchor to another |
| [DisplayName](https://ninjatrader.com/es/support/helpGuides/nt8/displayname.htm) | A string value which sets the name prefix used for all properties for a chart anchor |
| [DrawingTool](https://ninjatrader.com/es/support/helpGuides/nt8/drawingtool.htm) | The drawing tool which owns a chart anchor |
| [DrawnOnBar](https://ninjatrader.com/es/support/helpGuides/nt8/drawnonbar.htm) | Gets the current bar value that the chart anchor is drawn by a NinjaScript object. |
| [GetPoint()](https://ninjatrader.com/es/support/helpGuides/nt8/getpoint.htm) | Returns a chart anchor's data points. |
| [IsBrowsable](https://ninjatrader.com/es/support/helpGuides/nt8/isbrowsable.htm) | A bool value determining the anchor is visible on the UI. |
| [IsEditing](https://ninjatrader.com/es/support/helpGuides/nt8/isediting.htm) | A bool value determining the anchor is currently being edited |
| [IsNinjaScriptDrawn](https://ninjatrader.com/es/support/helpGuides/nt8/isninjascriptdrawn.htm) | Indicates if the chart anchor was drawn by a NinjaScript object |
| [IsXPropertiesVisible](https://ninjatrader.com/es/support/helpGuides/nt8/isypropertyvisibile.htm) | A bool value determining the X properties are visible on the UI |
| [IsYPropertyVisible](https://ninjatrader.com/es/support/helpGuides/nt8/isypropertyvisibile.htm) | A bool value determining the Y data value is visible on the UI |
| [MoveAnchor()](https://ninjatrader.com/es/support/helpGuides/nt8/moveanchor.htm) | Moves a Chart Anchor's x and y values from start point by a delta point amount. |
| [MoveAnchorX()](https://ninjatrader.com/es/support/helpGuides/nt8/moveanchorx.htm) | Moves an anchor x values from start point by a delta point amount |
| [MoveAnchorY()](https://ninjatrader.com/es/support/helpGuides/nt8/moveanchory.htm) | Moves an anchor y values from start point by a delta point amount |
| [Price](https://ninjatrader.com/es/support/helpGuides/nt8/price.htm) | Determines price value the chart anchor is drawn. |
| [SlotIndex](https://ninjatrader.com/es/support/helpGuides/nt8/barindex.htm) | Indicates the nearest bar slot where anchor is drawn. |
| [Time](https://ninjatrader.com/es/support/helpGuides/nt8/time.htm) | Determines date/time value the chart anchor is drawn. |
| [UpdateFromPoint()](https://ninjatrader.com/es/support/helpGuides/nt8/updatefrompoint.htm) | Updates an anchor's x and y values from a given point (in device pixels) |
| [UpdateXFromPoint()](https://ninjatrader.com/es/support/helpGuides/nt8/updatexfrompoint.htm) | Updates an anchor's X values from a given point (in device pixels) |
| [UpdateYFromPoint()](https://ninjatrader.com/es/support/helpGuides/nt8/updateyfrompoint.htm) | Updates an anchor's Y value from a given point (in device pixels) |

**Examples**

| ns | |
| --- | --- |
| public ChartAnchor MyAnchor { get; set; }   // declares the "MyAnchor" ChartAnchor object   public override IEnumerable<ChartAnchor> Anchors { get { return new[] { MyAnchor }; } } //adds the "MyAnchor" ChartAnchor object to a collection of anchors used to interact with your anchors   protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Description = @"Drawing tool example";     Name = "SampleDrawingTool";       MyAnchor = new ChartAnchor(); //creates a new instances of the ChartAnchor object     MyAnchor.IsEditing   = true;     MyAnchor.DrawingTool = this;     MyAnchor.IsBrowsable = false;   } }   public override void OnMouseUp(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {   if (DrawingState == DrawingState.Editing)   {     if (MyAnchor.IsEditing)     {         //if anchor is editing, update anchor point         dataPoint.CopyDataValues(MyAnchor);     }   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **CopyDataValues()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/displayname.htm) |

Definition  
Copies the ChartAnchor time and price values from on anchor to another.  This includes the [BarsAgo](https://ninjatrader.com/es/support/helpGuides/nt8/barsago.htm), [SlotIndex](https://ninjatrader.com/es/support/helpGuides/nt8/barindex.htm), [Time](https://ninjatrader.com/es/support/helpGuides/nt8/time.htm), [Price](https://ninjatrader.com/es/support/helpGuides/nt8/price.htm), and [DrawnOnBar](https://ninjatrader.com/es/support/helpGuides/nt8/drawnonbar.htm) values.  This method is useful for updating a chart anchor to a recent data point when the user interacts with the drawing chart anchor.

**Method Return Value**

This method does not return a value.

**Syntax**

<chartAnchor>.CopyDataValues(ChartAnchor toAnchor)

**Method Parameters**

|  |  |
| --- | --- |
| toAnchor | The ChartAnchor to copy |

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseMove(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {     // if the user is moving the draw object, copy the most recent dataPoint to MyAnchor   if (DrawingState == DrawingState.Moving)             dataPoint.CopyDataValues(Anchor); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **DisplayName** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/copydatavalues.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawingtool.htm) |

**Definition**

Sets the display name prefix used for all properties for a chart anchor.

**Property Value**

A string value that is used to identify the name for a corresponding anchor.  Default value is **null**.

**Syntax**

<ChartAnchor>.DisplayName

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {                 MyAnchor = new ChartAnchor();       MyAnchor.DisplayName = "MyChartAnchor";   }   } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **DrawingTool** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/displayname.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawnonbar.htm) |

**Definition**

The DrawingTool object which owns a chart anchor.

**Property Value**

A IDrawingTool object representing the owner of the chart anchor

**Syntax**

<ChartAnchor>.DrawingTool

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)      {  Name = "SampleDrawingTool";  MyAnchor = new ChartAnchor();  MyAnchor.DrawingTool = this; //NinjaTrader.NinjaScript.DrawingTools.SampleDrawingTool      }      else if (State == State.Configure)      {        } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **DrawnOnBar** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawingtool.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getpoint.htm) |

**Definition**

Gets the current bar value that the chart anchor is drawn by a NinjaScript object.  Please see the [Drawing](https://ninjatrader.com/es/support/helpGuides/nt8/drawing.htm) section for more information.

|  |
| --- |
| **Note**:  This value will **NOT** work on manually drawn objects.  This property is reserved for chart anchors which were drawn by another NinjaScript object (e.g, using a Draw method in an indicator).  For manually drawn objects, please see the [SlotIndex](https://ninjatrader.com/es/support/helpGuides/nt8/barindex.htm) property |

**Property Value**

A int value that value which the current bar the chart anchor is drawn.  This property is read-only.

**Syntax**

<ChartAnchor>.DrawnOnBar

**Examples**

| ns | |
| --- | --- |
| //Places text if high is 2419 and prints what bar the text was drawn on  if (High[0] == 2419)  {   Text myText = Draw.Text(this, @"Text " + CurrentBar, @"High is 2419" , 0, High[0]);   Print("Text is on bar " + myText.Anchor.DrawnOnBar);  } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **GetPoint()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawnonbar.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isbrowsable.htm) |

**Definition**

Returns a chart anchor's data point in device pixels

**Method Return Value**

A [Point](https://msdn.microsoft.com/en-us/library/system.drawing.point%28v=vs.110%29.aspx) structure; a point value in device pixels for a chart's given panel & scale

**Syntax**

<chartAnchor>.GetPoint(ChartControl chartControl, ChartPanel chartPanel, ChartScale, [bool pixelAlign])

**Method Parameters**

|  |  |
| --- | --- |
| chartControl | A [ChartControl](https://ninjatrader.com/es/support/helpGuides/nt8/chartcontrol.htm) representing the x-axis |
| chartPanel | A [ChartPanel](https://ninjatrader.com/es/support/helpGuides/nt8/chartpanel.htm) representing the a panel of the chart |
| chartScale | A [ChartScale](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale.htm) representing the y-axis |
| pixelAlign | An optional bool determining if the data point should be rounded to closest .5 pixel point |

**Examples**

| ns | |
| --- | --- |
| //gets the chart anchors data points  Point anchorPoint = MyAnchor.GetPoint(chartControl, chartPanel, chartScale); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **IsBrowsable** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getpoint.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isediting.htm) |

**Definition**

Determines if the anchor are visible on the UI.  When set to true, the anchors Y and X values can be viewed from the Drawing Objects properties.

**Property Value**

A bool value which when true will display the anchor data values from the drawing object properties; otherwise **false**.  Default value is **true.**

**Syntax**

<ChartAnchor>.IsBrowsable

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)      {  MyAnchor = new ChartAnchor();  MyAnchor.IsBrowsable = true;      }      else if (State == State.Configure)      {        } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **IsEditing** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isbrowsable.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isninjascriptdrawn.htm) |

**Definition**  
Determines if the anchor can be edited.

**Property Value**

A bool value which when true determines if the chart anchor is currently in a state it can be edited.  Default is **false**.

**Syntax**

<ChartAnchor>.IsEditing

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseDown(ChartControl *chartControl*, ChartPanel *chartPanel*, ChartScale *chartScale*, Point *point*)  {  if(DrawingState == DrawingState.Building)  {    // if drawing tool is currently editing, update to current mouse point  if(MyAnchor.IsEditing)  {  MyAnchor.UpdateFromPoint(point, chartControl, chartScale);    //set the anchor to disable editing when done updating  MyAnchor.IsEditing = false;  }  }  } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **IsNinjaScriptDrawn** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isediting.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isxpropertiesvisible.htm) |

**Definition**

Indicates if the chart anchor was drawn by a NinjaScript object (such as an indicator or strategy).

**Property Value**

A bool value which returns **true** of the object was drawn by other NinjaScript object; otherwise **false**.  This property is read-only.

**Syntax**

<ChartAnchor>.IsNinjaScriptDrawn

**Examples**

| ns | |
| --- | --- |
| //unlocks the NinjaScript drawn object and allows the user to modify the anchor, while the NinjaScript object still 'owns' the object protected override void OnBarUpdate() {     foreach(IDrawingTool dt in DrawObjects)         {           DrawingTools.Line sampleLine = dt as DrawingTools.Line;                       if (sampleLine != null && sampleLine.StartAnchor.IsNinjaScriptDrawn)           {               sampleLine.IsLocked = false;               Print(sampleLine.StartAnchor.ToString());           }         } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **IsYPropertyVisibile** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isxpropertiesvisible.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/moveanchor.htm) |

**Definition**

Indicates the anchor's Y properties are visible on the UI.  When set to true, the Y values can be viewed from the Drawing Objects properties.

**Property Value**

A bool value which when true will display the anchor's Y (price) data values from the drawing object properties; otherwise **false**.  Default value is **true.**

**Syntax**

<ChartAnchor>.IsYPropertyVisibile

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)      {  MyAnchor = new ChartAnchor();  MyAnchor.IsYPropertyVisibile = true;      }      else if (State == State.Configure)      {        } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **IsYPropertyVisibile** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isxpropertiesvisible.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/moveanchor.htm) |

**Definition**

Indicates the anchor's Y properties are visible on the UI.  When set to true, the Y values can be viewed from the Drawing Objects properties.

**Property Value**

A bool value which when true will display the anchor's Y (price) data values from the drawing object properties; otherwise **false**.  Default value is **true.**

**Syntax**

<ChartAnchor>.IsYPropertyVisibile

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)      {  MyAnchor = new ChartAnchor();  MyAnchor.IsYPropertyVisibile = true;      }      else if (State == State.Configure)      {        } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **MoveAnchor()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isypropertyvisibile.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/moveanchorx.htm) |

**Definition**

Moves a Chart Anchor's x and y values from start point by a delta point amount.

**Method Return Value**

This method does not return a value.

**Syntax**

<ChartAnchor>.MoveAnchor(ChartAnchor startDataPoint, ChartAnchor deltaDataPoint, ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, DrawingTool drawingTool)

**Method Parameters**

|  |  |
| --- | --- |
| startPoint | The chart anchor's original starting location value represented by a point structure |
| startDataPoint | A chart anchor's original starting location value represented by a chart anchor |
| deltaPoint | The chart anchor's new location value to be updated represented by a point structure |
| deltaDataPoint | The chart anchor's new location value to be udpated represened by a chart anchor |
| chartControl | A ChartControl representing the x-axis |
| chartScale | A ChartScale representing the y-axis |
| chartPanel | A ChartPanel representing the the panel for the chart |
| drawingTool | The drawing tool which owns the chart anchor to be moved (usually this). |

**Examples**

| ns | |
| --- | --- |
| //move the chart anchors x and y values MyAnchor.MoveAnchor(lastPoint, newPoint, chartControl, chartPanel, chartScale, this); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **MoveAnchorX()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/moveanchor.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/moveanchory.htm) |

**Definition**

Moves an anchor's x value from start point by a delta point amount.

**Method Return Value**

This method does not return a value.

**Syntax**

<ChartAnchor>.MoveAnchorX*(*Point*startPoint,*Point*deltaPoint,*ChartControl*chartControl,*ChartPanel*chartPanel,*ChartScale*chartScale)*

**Method Parameters**

|  |  |
| --- | --- |
| startPoint | The chart anchor's original starting point value |
| deltaPoint | The chart anchor's new point value to be updated |
| chartControl | A ChartControl representing the x-axis |
| chartScale | A ChartScale representing the y-axis |

**Examples**

| ns | |
| --- | --- |
| //move only the chart anchors x (bar/time) value  MyAnchor.MoveAnchorX(lastPoint, newPoint, chartControl, chartScale); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **MoveAnchorY()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/moveanchorx.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/price.htm) |

**Definition**

Moves an anchor's y value from start point by a delta point amount.

**Method Return Value**

This method does not return a value.

**Syntax**

<ChartAnchor>.MoveAnchorY(Point startPoint, Point deltaPoint, ChartControl chartControl, ChartScale chartScale)

**Method Parameters**

|  |  |
| --- | --- |
| startPoint | The chart anchor's original starting point value |
| deltaPoint | The chart anchor's new point value to be updated |
| chartControl | A ChartControl representing the x-axis |
| chartScale | A ChartScale representing the y-axis |

**Examples**

| ns | |
| --- | --- |
| //move only the chart anchors Y (price) value  MyAnchor.MoveAnchorY(lastPoint, newPoint, chartControl, chartPanel, chartScale); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **Price** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/moveanchory.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/barindex.htm) |

**Definition**

Determines price value the chart anchor is drawn.

**Property Value**

An double value representing a price value

**Syntax**

<ChartAnchor>.Price

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseDown(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, Point point) {   Print(MyAnchor.Price); // prints the Y axis data point of the chart anchor   // 1999.25 } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **SlotIndex** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/price.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/time.htm) |

**Definition**

Indicates the nearest bar slot value where anchor is drawn on a chart.  In a single series chart there will always be equal number of slots and bars, however for multi-series charts there may be additonal slots compared to the bar series your drawing tool resides.

**Property Value**

An double value representing the current bar.

|  |
| --- |
| **Note**:  The bar index value is represented as a double as it is possible (and likely) that a given chart anchor is drawn between bars (i.e., if a user draws the tool with snap mode disabled) |

**Syntax**

ChartAnchor.SlotIndex

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseDown(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {   Print(MyAnchor.SlotIndex); // prints the nearest current bar value   //4502.02734375 } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **Time** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/barindex.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/updatefrompoint.htm) |

**Definition**

Determines date/time value the chart anchor is drawn.

**Property Value**

An DateTime value representing a time value

**Syntax**

<ChartAnchor>.Time

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseDown(ChartControl *chartControl*, ChartPanel *chartPanel*, ChartScale *chartScale*, Point *point*)  {  Print(MyAnchor.Time); // prints the X axis datetime of the chart anchor  // 8/26/2014 6:55:00 PM  } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **UpdateFromPoint()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/time.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/updatexfrompoint.htm) |

**Definition**

Updates an anchor's x and y values from a given point (in device pixels).

**Method Return Value**

This method does not return a value.

**Syntax**

<ChartAnchor>.UpdateFromPoint(Point point, ChartControl chartControl, ChartScale chartScale)

**Method Parameters**

|  |  |
| --- | --- |
| point | The chart anchor's point value to be updated |
| chartControl | A ChartControl representing the x-axis |
| chartScale | A ChartScale representing the y-axis |

**Examples**

| ns | |
| --- | --- |
| //set the chart anchors x and y point value  MyAnchor.UpdateFromPoint(point, chartControl, chartScale); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **UpdateXFromPoint()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/updatefrompoint.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/updateyfrompoint.htm) |

**Definition**

Updates an anchor's X value from a given point (in device pixels).

**Method Return Value**

This method does not return a value.

**Syntax**

<ChartAnchor>.UpdateXFromPoint(Point *point*, ChartControl *chartControl*, ChartScale *chartScale*)

**Method Parameters**

|  |  |
| --- | --- |
| point | The chart anchor's point value to be updated |
| chartControl | A ChartControl representing the x-axis |
| chartScale | A ChartScale representing the y-axis |

**Examples**

| ns | |
| --- | --- |
| //set the chart anchors x point value  MyAnchor.UpdateXFromPoint(point, chartControl, chartScale); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) > [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) >  **UpdateYFromPoint()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/updatexfrompoint.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/converttoverticalpixels.htm) |

**Definition**

Updates an anchor's Y value from a given point (in device pixels).

**Method Return Value**

This method does not return a value.

**Syntax**

<ChartAnchor>.UpdateYFromPoint(Point point, ChartScale chartScale)

**Method Parameters**

|  |  |
| --- | --- |
| point | The chart anchor's point value to be updated |
| chartScale | A ChartScale representing the y-axis |

**Examples**

| ns | |
| --- | --- |
| //set the chart anchors x point value  MyAnchor.UpdateYFromPoint(point, chartScale); | |
| **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()](https://ninjatrader.com/es/support/helpGuides/nt8/draw_regressionchannel.htm) >  **RegressionChannel** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_regressionchannel.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/draw_regressionchannel.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/draw_riskreward.htm) |

**Definition**

Represents an interface that exposes information regarding a Regression Channel [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 |
| RegressionStroke | The [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the middle line of the object |
| LowerChannelStroke | The [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the lower line of the object |
| UpperChannelStroke | The [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm) object used to draw the upper line of the object |
| PriceType | Possible values are:   PriceType.Close  PriceType.High  PriceType.Low  PriceType.Median  PriceType.Open  PriceType.Typical |
| ChannelType | An enum value representing if the object will use standard deviations calculations for the upper/lower lines.  Possible values are    •RegressionChannelType.Segment,  •RegressionChannelType.StandardDeviation |
| ExtendLeft | A bool value representing if the object will extend to the left |
| ExtendRight | A bool value representing if the object will extend to the right |
| StandardDeviationLowerDistance | A double value representing the standard deviation distance to the lower line |
| StandardDeviationUpperDistance | A double value representing the standard deviation distance to the upper line |

**Example**

| ns |
| --- |
| // Instantiate a RegressionChannel object NinjaTrader.NinjaScript.DrawingTools.RegressionChannel myRegChan = Draw.RegressionChannel(this, "tag1", 10, 0, Brushes.Blue);   // Change the object's PriceType myRegChan.PriceType = PriceType.Median; |

|  |  |
| --- | --- |
| **Note**: To differentiate between DrawingTools.RegressionChannel and Indicators.RegressionChannel when assigning a RegressionChannel 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) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **AddPastedOffset()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/anchors.htm) |

**Definition**

A [virtual method](https://msdn.microsoft.com/en-us/library/9fkccyh4.aspx) which is called every time a DrawingTool is copied and pasted to a chart.  The default behavior will offset the chart anchors price value down by 1, percent. However, this behavior can be overridden for your custom drawing tool if desired.

**Method Return Value**

This method does not return a value

**Syntax**

You must override this method using the following syntax:

public override void AddPastedOffset(ChartPanel panel, ChartScale chartScale)  
{  
   
}

**Method Parameters**

|  |  |
| --- | --- |
| panel | A ChartPanel representing the the panel for the chart |
| chartScale | A ChartScale representing the Y-axis |

**Examples**

| ns | |
| --- | --- |
| public override void AddPastedOffset(ChartPanel chartPanel, ChartScale chartScale) {         foreach (ChartAnchor anchor in Anchors)   {     //bump each anchor 1 minute to the right     DateTime tmpTime = anchor.Time;     anchor.Time = tmpTime.AddMinutes(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) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **Anchors** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/addpastedoffset.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/attachedto.htm) |

**Definition**

Returns a custom collection of ChartAnchors which will represent various points of the drawing tool.

|  |
| --- |
| **Note**:  You must declare this property with the chart anchors used in the drawing tool which you plan on using for iteration.  Doing so will expose a simple enumerator which will allow you to to iterate over the chart anchors in which have been defined in this interface. |

**Property Value**

A virtual [IEnumerable](https://msdn.microsoft.com/en-us/library/9eekhta0%28v=vs.110%29.aspx) interface consisting of [ChartAnchors](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm)

**Syntax**

You must override this property using the following syntax:

public override IEnumerable<ChartAnchor> Anchors  
{  
   
}

**Examples**

| ns | |
| --- | --- |
| //defines the chart anchors used for the drawing tool public ChartAnchor      StartAnchor    { get; set; } public ChartAnchor      MiddleAnchor   { get; set; } public ChartAnchor      EndAnchor      { get; set; }   //create a collection of chart anchors used for a simple iteration public override IEnumerable<ChartAnchor> Anchors {   get   {     return new[] { StartAnchor, MiddleAnchor, EndAnchor };   } }   //setup our chart anchor instances and assign a display name to each protected override void OnStateChange() {   if (State == State.SetDefaults)   {            Name                  = "My Drawing Tool";        StartAnchor           = new ChartAnchor();      MiddleAnchor          = new ChartAnchor();      EndAnchor             = new ChartAnchor();        StartAnchor.DisplayName    = "My Start Anchor";      MiddleAnchor.DisplayName   = "My Middle Anchor";      EndAnchor.DisplayName      = "My End Anchor";           } }   //for each render pass, print out the display name of the chart anchors protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {     foreach (ChartAnchor anchor in Anchors)   {      Print(anchor.DisplayName);   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **AttachedTo** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/anchors.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) |

**Definition**

An object which holds information regarding where the drawing tool is attached.

**Available Properties**

|  |  |
| --- | --- |
| AttachedToType | An enum representing the type of object the drawing to is attached.  Possible values are:  •Bars - The chart bars of the parent chart  •GlobalInstrument - The bars of an instrument crossed all charts  •Indicator - A NinjaScript indicator  •Strategy - A NinjaScript strategy |
| ChartObject | A ChartObject interface such an indicator, strategy, chart bars |
| DisplayName | A string value indicating the name of the object the drawing tool is attached |
| Instrument | The [Instrument](https://ninjatrader.com/es/support/helpGuides/nt8/instrument.htm) that the drawing tool is attached |

**Syntax**

AttachedTo

**Examples**

|  |  |
| --- | --- |
| ns | |
| if (AttachedTo.AttachedToType == AttachedToType.Indicator)   // do something | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **ConvertToVerticalPixels()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/updateyfrompoint.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/createanchor.htm) |

**Definition**

Used to convert the cursor position (pixels) to device pixels represented on the Y axis of the chart.  This method would only be needed if the value you are given is provided in WPF pixel point (such as the data point used in OnMouseDown), but you would need the value in the chart's rendered pixels.  This is useful when handling drawing tools and charts which would have multiple chart panels.

**Method Return Value**

An int value representing the converted value in device pixels

**Syntax**

ConvertToVerticalPixels(ChartControl chartControl, ChartPanel chartPanel, double wpfY)

**Method Parameters**

|  |  |
| --- | --- |
| chartControl | A ChartControl representing the x-axis |
| chartPanel | A ChartPanel representing the the panel for the chart |
| wpfY | A double value which needs to be converted |

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseDown(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {   //get chart anchors data point when mouse is clicked   Point myPoint = dataPoint.GetPoint(chartControl, chartPanel, chartScale);     Print("before convert: " + myPoint.Y); //before convert: 630.5     //convert the data point to device pixels   double yPixel = ConvertToVerticalPixels(chartControl, chartPanel, myPoint.Y);       Print("after convert: " + yPixel); //after convert: 1108 } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **CreateAnchor()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/converttoverticalpixels.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/displayonchartsmenus.htm) |

**Definition**

Used to create a new chart anchor at a specified mouse point.

**Method Return Value**

A new [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) at a specified point in device pixels.

**Syntax**

CreateAnchor(Point point, ChartControl chartControl, ChartScale chartScale)

**Method Parameters**

|  |  |
| --- | --- |
| point | A Point in device pixels representing the current mouse cursor position |
| chartControl | A ChartControl representing the x-axis |
| chartScale | A ChartScale representing the y-axis |

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseDown(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {   // get the point where the mouse was clicked   Point myPoint = dataPoint.GetPoint(chartControl, chartPanel, chartScale);     // create an anchor at that point   ChartAnchor MyAnchor = CreateAnchor(myPoint, chartControl, chartScale);     Print(MyAnchor.Time); // 3/16/2015 8:18:48 AM } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **CreateAnchor()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/converttoverticalpixels.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/displayonchartsmenus.htm) |

**Definition**

Used to create a new chart anchor at a specified mouse point.

**Method Return Value**

A new [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) at a specified point in device pixels.

**Syntax**

CreateAnchor(Point point, ChartControl chartControl, ChartScale chartScale)

**Method Parameters**

|  |  |
| --- | --- |
| point | A Point in device pixels representing the current mouse cursor position |
| chartControl | A ChartControl representing the x-axis |
| chartScale | A ChartScale representing the y-axis |

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseDown(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {   // get the point where the mouse was clicked   Point myPoint = dataPoint.GetPoint(chartControl, chartPanel, chartScale);     // create an anchor at that point   ChartAnchor MyAnchor = CreateAnchor(myPoint, chartControl, chartScale);     Print(MyAnchor.Time); // 3/16/2015 8:18:48 AM } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **DisplayOnChartsMenus** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/createanchor.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/dispose.htm) |

**Definition**

Determines if the drawing tool displays in the chart's drawing tool menus.

**Property Value**

A bool value, when **true** the drawing tool will be created on the chart's drawing tool menu; otherwise **false**.  Default value is **true**.

**Syntax**

DisplayOnChartsMenus

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name                 = @"My Drawing Tool";     DisplayOnChartsMenus = 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) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **Dispose()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/displayonchartsmenus.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawingstate.htm) |

**Definition**

Releases any device resources used for the drawing tool.

**Method Return Value**

This method does not return a value

**Syntax**

Dispose()

**Method Parameters**

This method does not accept any parameters

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name                 = @"My Drawing Tool";         }     else if (State == State.Terminated)     Dispose(); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **DrawingState** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/dispose.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/drawnby.htm) |

**Definition**

Represents the current state of the drawing tool to perform various actions, such as building, editing, or moving.

**Property Values**

An enum representing the current state of the drawing tool.  Possible values are:

|  |  |
| --- | --- |
| DrawingState.Building | The initial state when a drawing tool is first being drawn, allowing for the anchors to be set for the drawing. |
| DrawingState.Editing | Allows for changing the values of any of the drawing tools anchors |
| DrawingState.Normal | The drawing tool is normal on the chart and is not in a state to allow for changes. |
| DrawingState.Moving | The entire drawing tool to be moved by a user. |

**Syntax**

DrawingState

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseDown(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, Point point) {   switch(DrawingState)   {                             case DrawingState.Normal:         DrawingState = DrawingState.Editing; // set state to allow editing         break;             case DrawingState.Editing:         // do your edits here         break;     case DrawingState.Moving:         return; // don't allow move whe editing               }       } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **DrawnBy** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawingstate.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getattachedtochartbars.htm) |

**Definition**

Represents the NinjaScript object which created the drawing object

**Property Value**

The NinjaScript object which created the drawing tool; this value will be null if drawn by a user.

**Syntax**

DrawnBy

**Examples**

| ns | |
| --- | --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {         // if the drawing tool was not created by a user,   // print the name of the object that it was created         if(!IsUserDrawn)   Print(DrawnBy.Name); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **GetAttachedToChartBars()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/drawnby.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getclosestanchor.htm) |

**Definition**

Returns information which relate to the underlying bars series in which the drawing tool is attached.  If the drawing tool is attached to an indicator rather than a bar series, the indicator's bars series used for input will be returned.

|  |
| --- |
| **Note**: For drawing tools made global, this method will not be returning meaningful values - since those are not attached to a specific bars series |

**Method Return Value**

A [ChartBars](https://ninjatrader.com/es/support/helpGuides/nt8/chartbars.htm) object

**Syntax**

GetAttachedToChartBars()

**Method Parameters**

This method does not accept any parameters

**Examples**

| ns | |
| --- | --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {     // get the attached chart bars   ChartBars myBars = GetAttachedToChartBars();     Print(myBars.Bars.ToChartString()); // NQ 03-15 (1 Minute) } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **GetClosestAnchor()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getattachedtochartbars.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getcursor.htm) |

**Definition**

Returns the closest chart anchor within a specified maximum distance from the mouse cursor.

**Method Return Value**

This method returns an existing [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm)

**Syntax**

GetClosestAnchor(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, double maxDist, Point point)

**Method Parameters**

|  |  |
| --- | --- |
| chartControl | A ChartControl representing the x-axis |
| chartPanel | A ChartPanel representing the the panel for the chart |
| chartScale | A ChartScale representing the y-axis |
| maxDist | A double value representing the cursor's sensitivity used to detect the nearest anchor |
| point | A Point in device pixels representing the current mouse cursor position |

**Examples**

| ns | |
| --- | --- |
| public override Cursor GetCursor(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, Point point) {           // get the closest anchor to where the user has clicked   ChartAnchor   closest = GetClosestAnchor(chartControl, chartPanel, chartScale, 10, point);     if (closest != null)   {     // set cursor to indicate that it can be moved     return Cursors.SizeNWSE;   }     // otherwise set cursor back to arrow   else return Cursors.Arrow; } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **GetCursor()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getclosestanchor.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getselectionpoints.htm) |

**Definition**

An event driven method which is called when a chart object is selected.  This method can be used to change the cursor image used in various states.

**Method Return Value**

This method returns a [Cursor](https://msdn.microsoft.com/en-us/library/system.windows.forms.cursor(v=vs.110).aspx) used to paint the mouse pointer.

**Syntax**  
You must override the method in your Drawing Tool with the following syntax:

public override Cursor GetCursor(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, Point point)

{

}

**Method Parameters**

|  |  |
| --- | --- |
| chartControl | A [ChartControl](https://ninjatrader.com/es/support/helpGuides/nt8/chartcontrol.htm) representing the x-axis |
| chartPanel | A [ChartPanel](https://ninjatrader.com/es/support/helpGuides/nt8/chartpanel.htm) representing the the panel for the chart |
| chartScale | A [ChartScale](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale.htm) representing the y-axis |
| point | A Point in device pixels representing the current mouse cursor position |

**Examples**

| ns | |
| --- | --- |
| public override Cursor GetCursor(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, Point point) {   switch (DrawingState)   {     //when drawing, display the cursor as a pen     case DrawingState.Building:   return Cursors.Pen;       // when moving, display a four-headed sizing cursor     case DrawingState.Moving:   return Cursors.SizeAll;       default: return Cursors.Pen;   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **GetSelectionPoints()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getcursor.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/icon_drawingtool.htm) |

**Definition**

Returns the chart object's data points where the user can interact.   These points are used to visually indicate that the chart object is selected and allow the user to manipulate the chart object.  This method is only called when [IsSelected](https://ninjatrader.com/es/support/helpGuides/nt8/isselected.htm) is set to true.

**Method Return Value**

A collection of [Points](https://msdn.microsoft.com/en-us/library/system.drawing.point%28v=vs.110%29.aspx) representing the x- and y-coordinates of the chart object.

**Syntax**  
You must override the method using the following syntax:

public override Point[] GetSelectionPoints(ChartControl chartControl, ChartScale chartScale)  
{  
   
}

**Method Parameters**

|  |  |
| --- | --- |
| chartControl | A [ChartControl](https://ninjatrader.com/es/support/helpGuides/nt8/chartcontrol.htm) representing the x-axis |
| chartScale | A [ChartScale](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale.htm) representing the y-axis |

**Examples**

| ns | |
| --- | --- |
| public override Point[] GetSelectionPoints(ChartControl *chartControl*, ChartScale *chartScale*)  {     ChartPanel chartPanel = chartControl.ChartPanels[chartScale.PanelIndex];    // get the anchor point to be displayed on the drawing tool   Point anchorPoint = Anchor.GetPoint(chartControl, chartPanel, chartScale, false);           return new[] { anchorPoint } ;  } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **Icon** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getselectionpoints.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/ignoressnapping.htm) |

**Definition**

The shape which displays next to the Drawing Tool menu item.  Since this is a standard object, any type of icon can be used (unicode characters, custom image file resource, geometry path, etc). For more information on using images to create icons, see the [Using Images with Custom Icons](https://ninjatrader.com/es/support/helpGuides/nt8/using_images_and_geometry_with_custom_icons.htm) page.

|  |
| --- |
| Note: When using UniCode characters, first ensure that the desired characters exist in the icon pack for the font family used in NinjaTrader. |

**Property Value**

A generic virtual object representing the drawing tools menu icon.  This property is read-only.

**Syntax**

You must override this property using the following syntax:

public override object Icon

**Examples**

| ns | |
| --- | --- |
| public override object Icon {           get   {     //use a unicode character as our string which will render an arrow     string uniCodeArrow = "\u279A";                 return uniCodeArrow;   }   } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **IgnoresSnapping** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/icon_drawingtool.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/ignoresuserinput.htm) |

**Definition**  
Determines if the drawing tool chart anchor's will use the chart's **Snap Mode** mouse coordinates.

**Property Value**

A bool value which when **true** the drawing tool ignores snapping; otherwise **false**.  Default is set to **false**.

**Syntax**

IgnoresSnapping

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)      {           IgnoresSnapping = true; // Set this to true to receive non-snapped mouse coordinates      }      else if (State == State.Configure)      {        } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **IgnoresUserInput** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ignoressnapping.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isattachedtoninjascript.htm) |

**Definition**  
Determines if the drawing tool can be clicked on by the user.

**Property Value**

A bool value which wen **true**if the drawing tool cannot not be interacted with by a user; otherwise **false**.  Default is set to **false**.

**Syntax**

IgnoresUserInput

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)      {      IgnoresUserInput = true; // Set this to true to make the drawing object non-interactive      }      else if (State == State.Configure)      {        } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **IsAttachedToNinjaScript** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ignoresuserinput.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isglobaldrawingtool.htm) |

**Definition**

Indicates if the drawing tool is currently [attached to](https://ninjatrader.com/es/support/helpGuides/nt8/attachedto.htm) a NinjaScript object (such an indicator or a strategy).

**Property Value**

A bool value which when **true**if the drawing tool is attached to a NinjaScript object; otherwise **false**.  This property is read-only.

**Syntax**

IsAttachedToNinjaScript

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseMove(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {     // do not interact if drawn by an indicator or strategy   if (IsAttachedToNinjaScript)     return; } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **IsGlobalDrawingTool** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isattachedtoninjascript.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/islocked.htm) |

**Definition**

Indicates if the drawing tool is currently set as a Global Drawing object. Global draw objects display on any chart which matches the parent chart's underlying instrument.

**Property Value**

A bool value which returns **true** if the drawing tool is currently attached as a global drawing object; otherwise **false**.

**Syntax**

IsGlobalDrawingTool

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseMove(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {     // do not interact if attached to global chart   if (IsGlobalDrawingTool)     return; } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **IsLocked** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isglobaldrawingtool.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isuserdrawn.htm) |

**Definition**  
Determines if the drawing tool should be be locked in place.  This property can be set either manually through the UI or explicitly through code.

**Property Value**

A bool value which when **true**if the drawing tool is locked; otherwise **false**.  Default is set to **false**.

|  |
| --- |
| **Note**:For Drawing tools which are drawn by an indicator or strategy, this property will default to **true**. |

**Syntax**

IsLocked

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseMove(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, Point point) {   if (IsLocked) //if the object is locked, do not attempt to move     return; } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **IsUserDrawn** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/islocked.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/onbarschanged.htm) |

**Definition**  
Indicates if the drawing tool was manually drawn by a user as opposed to programmatically drawn by a NinjaScript object (such as an indicator or strategy).

**Property Value**

A bool value which when **true**if the draw object was manually drawn ; otherwise **false.**This property is read-only

**Syntax**

IsUserDrawn

**Examples**

| ns | |
| --- | --- |
| if (IsUserDrawn)  {  // do something only if the object was drawn manually  } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **OnBarsChanged()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isuserdrawn.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/onmousedown.htm) |

**Definition**

An event driven method which is called any time the underlying bar series have changed for the chart where the drawing tool resides.  For example if a user has changed the primary instrument or the time frame of the bars used on the chart.

**Method Return Value**

This method does not return a value

**Syntax**

You must override this method using the following syntax:

public override void OnBarsChanged()  
{  
   
}

**Method Parameters**

This method does not accept any parameters

**Examples**

| ns | |
| --- | --- |
| public override void OnBarsChanged() {    //bars have change, do something         } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **OnMouseDown()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onbarschanged.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/onmousemove.htm) |

**Definition**

An event driven method which is called any time the mouse pointer over the chart control has the mouse button pressed.

**Method Return Value**

This method does not return a value.

|  |
| --- |
| **Note**:  For a combined single click operation, i.e. mouse down click, move and release the dataPoint reported will always be the initial starting one. |

**Syntax**  
You must override the method in your Drawing Tool with the following syntax.

public override void OnMouseDown(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint)  
{  
   
}

**Method Parameters**

|  |  |
| --- | --- |
| chartControl | A [ChartControl](https://ninjatrader.com/es/support/helpGuides/nt8/chartcontrol.htm) representing the x-axis |
| chartPanel | A [ChartPanel](https://ninjatrader.com/es/support/helpGuides/nt8/chartpanel.htm) representing the the panel for the chart |
| chartScale | A [ChartScale](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale.htm) representing the y-axis |
| dataPoint | A [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) representing a point where the user clicked |

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseDown(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {   switch (DrawingState)   {     case DrawingState.Building:         dataPoint.CopyDataValues(Anchor);         Anchor.IsEditing   = false;         DrawingState     = DrawingState.Normal;         IsSelected         = false;         break;     case DrawingState.Normal:         // make sure they clicked near us. use GetCursor incase something has more than one point         Point point = dataPoint.GetPoint(chartControl, chartPanel, chartScale);         if (GetCursor(chartControl, chartPanel, chartScale, point) != null)           DrawingState = DrawingState.Moving;         else           IsSelected = false;         break;   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **OnMouseDown()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onbarschanged.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/onmousemove.htm) |

**Definition**

An event driven method which is called any time the mouse pointer over the chart control has the mouse button pressed.

**Method Return Value**

This method does not return a value.

|  |
| --- |
| **Note**:  For a combined single click operation, i.e. mouse down click, move and release the dataPoint reported will always be the initial starting one. |

**Syntax**  
You must override the method in your Drawing Tool with the following syntax.

public override void OnMouseDown(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint)  
{  
   
}

**Method Parameters**

|  |  |
| --- | --- |
| chartControl | A [ChartControl](https://ninjatrader.com/es/support/helpGuides/nt8/chartcontrol.htm) representing the x-axis |
| chartPanel | A [ChartPanel](https://ninjatrader.com/es/support/helpGuides/nt8/chartpanel.htm) representing the the panel for the chart |
| chartScale | A [ChartScale](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale.htm) representing the y-axis |
| dataPoint | A [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) representing a point where the user clicked |

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseDown(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {   switch (DrawingState)   {     case DrawingState.Building:         dataPoint.CopyDataValues(Anchor);         Anchor.IsEditing   = false;         DrawingState     = DrawingState.Normal;         IsSelected         = false;         break;     case DrawingState.Normal:         // make sure they clicked near us. use GetCursor incase something has more than one point         Point point = dataPoint.GetPoint(chartControl, chartPanel, chartScale);         if (GetCursor(chartControl, chartPanel, chartScale, point) != null)           DrawingState = DrawingState.Moving;         else           IsSelected = false;         break;   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **OnMouseUp()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onmousemove.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/supportsalerts.htm) |

**Definition**

An event driven method is called any time the mouse pointer is over the chart control and a mouse button is being released.

**Method Return Value**

This method does not return a value

|  |
| --- |
| **Note**:  For a combined single click operation, i.e. mouse down click, move and release the dataPoint reported will always be the initial starting one. |

**Syntax**  
You must override the method with the following syntax.

public override void OnMouseUp(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint)  
{  
   
}

**Method Parameters**

|  |  |
| --- | --- |
| chartControl | A [ChartControl](https://ninjatrader.com/es/support/helpGuides/nt8/chartcontrol.htm) representing the x-axis |
| chartPanel | A [ChartPanel](https://ninjatrader.com/es/support/helpGuides/nt8/chartpanel.htm) representing the the panel for the chart |
| chartScale | A [ChartScale](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale.htm) representing the y-axis |
| dataPoint | A [ChartAnchor](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm) representing a point where the user is releasing the mouse |

**Examples**

| ns | |
| --- | --- |
| public override void OnMouseUp(ChartControl chartControl, ChartPanel chartPanel, ChartScale chartScale, ChartAnchor dataPoint) {   //when the user releases the mouse, ensure the drawing state is set to normal   if (DrawingState == DrawingState.Editing || DrawingState == DrawingState.Moving)     DrawingState = DrawingState.Normal; } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **SupportsAlerts** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onmouseup.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/zordertype.htm) |

**Definition**

Determines if the drawing tool can be used for manually configured alerts through the UI.

**Property Value**

A bool which when **true** determines that user can setup an alert based off this drawing tool;  otherwise **false**.

|  |
| --- |
| **Note**:  This property is **false** by default and **MUST** be overridden upon initialization to allow for manually configured alerts.  You cannot set this during run-time. |

**Syntax**

SupportsAlerts

You may choose to override this property using the following syntax:

public override bool SupportsAlerts

**Examples**

| ns | |
| --- | --- |
| public override bool SupportsAlerts { get { return true; } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **SupportsAlerts** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onmouseup.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/zordertype.htm) |

**Definition**

Determines if the drawing tool can be used for manually configured alerts through the UI.

**Property Value**

A bool which when **true** determines that user can setup an alert based off this drawing tool;  otherwise **false**.

|  |
| --- |
| **Note**:  This property is **false** by default and **MUST** be overridden upon initialization to allow for manually configured alerts.  You cannot set this during run-time. |

**Syntax**

SupportsAlerts

You may choose to override this property using the following syntax:

public override bool SupportsAlerts

**Examples**

| ns | |
| --- | --- |
| public override bool SupportsAlerts { get { return true; } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **ZOrderType** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/supportsalerts.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/import_type.htm) |

**Definition**

Determines the order in which the drawing tool will be rendered. This will help control the [ZOrder](https://ninjatrader.com/es/support/helpGuides/nt8/chart_zorder.htm) index between chart objects

**Property Value**

An enum determining the drawing tool's ZOrder type.  Possible values are:

|  |  |
| --- | --- |
| DrawingToolZOrder.Normal | Default behavior, drawing tools are rendered as they appear in the [ZOrder](https://ninjatrader.com/es/support/helpGuides/nt8/chart_zorder.htm) index |
| DrawingToolZOrder.AlwaysDrawnFirst | Ensures the drawing tool is always the first to be rendered |
| DrawingToolZOrder.AlwaysDrawnLast | Ensures the drawing tool is always the last object to be rendered |

**Syntax**

ZOrderType

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name               = @"My Drawing Tool";               // always draw this last     ZOrderType           = DrawingToolZOrder.AlwaysDrawnLast;   }   else if (State == State.Configure)   {   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Drawing Tool](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) >  **ZOrderType** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/supportsalerts.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/drawing_tools.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/import_type.htm) |

**Definition**

Determines the order in which the drawing tool will be rendered. This will help control the [ZOrder](https://ninjatrader.com/es/support/helpGuides/nt8/chart_zorder.htm) index between chart objects

**Property Value**

An enum determining the drawing tool's ZOrder type.  Possible values are:

|  |  |
| --- | --- |
| DrawingToolZOrder.Normal | Default behavior, drawing tools are rendered as they appear in the [ZOrder](https://ninjatrader.com/es/support/helpGuides/nt8/chart_zorder.htm) index |
| DrawingToolZOrder.AlwaysDrawnFirst | Ensures the drawing tool is always the first to be rendered |
| DrawingToolZOrder.AlwaysDrawnLast | Ensures the drawing tool is always the last object to be rendered |

**Syntax**

ZOrderType

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name               = @"My Drawing Tool";               // always draw this last     ZOrderType           = DrawingToolZOrder.AlwaysDrawnLast;   }   else if (State == State.Configure)   {   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [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) > [Educational Resources](https://ninjatrader.com/es/support/helpGuides/nt8/educational_resources.htm) >  **Using SharpDX for Custom Chart Rendering** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/using_images_and_geometry_with_custom_icons.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_brushes.htm) |

**Understanding the SharpDX .NET Library**

NinjaTrader Chart objects (such as Indicators, Strategies, DrawingTools, ChartStyles) implement an [OnRender()](https://ninjatrader.com/es/support/helpGuides/nt8/onrender.htm) method aimed to render custom lines, shapes, and text to the chart.  To achieve the level of performance required to keep up with market data events, NinjaTrader uses a 3rd-party open-source .NET library named [SharpDX](http://sharpdx.org/).  This 3rd party library provides a C# wrapper for the powerful [Microsoft DirectX API](https://msdn.microsoft.com/en-us/library/windows/desktop/ee663274(v=vs.85).aspx) used for graphics processing and known for its hardware-accelerated performance, including 2D vector and text layout graphics used for **NinjaTrader Chart Rendering**.  The SharpDX/DirectX library is extensive, although NinjaTrader only uses a handful of namespaces and classes, which are documented as a guide in this reference.  In addition to this educational resource, we have also compiled a more focused collection of [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) resources to help you learn the **SharpDX** concepts used in **NinjaTrader Chart Rendering**.

|  |
| --- |
| **Tips**:  1.There are several pre-installed examples of **OnRender()** and **SharpDX** objects used in the **NinjaTrader.Custom** project.  For starters, please look at the **SampleCustomRender** indicator file  2.Although not entirely identical, the **SharpDX** wrapper is designed to resemble **System.Drawing** namespace; experienced GDI developers will be familiar with concepts discussed in this section.  3.Microsoft provides various [DirectX Programming Guides](https://msdn.microsoft.com/en-us/library/dd372337(v=vs.85).aspx) aimed to educate users with the underlying**C++ DirectX API**.  While **SharpDX (C#)** syntax is different, you may find these guides helpful for understanding **SharpDX** concepts not offered by this guide. |

There are three main **SharpDX** namespaces you need to be familiar with:

|  |  |
| --- | --- |
| [SharpDX](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx.htm) | Contains basic objects used by SharpDX. |
| [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) | Contains objects used for rendering for 2D geometry, bitmaps, and text. |
| [SharpDX.DirectWrite](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite.htm) | Contains objects used for text rendering |

The rest of this page will help you navigate the fundamental concepts needed to achieve custom rendering to your charts.

tog_minus        [SharpDX Vectors and Charting Coordinates](javascript:HMToggle('toggle','SharpDXVectorsAndChartingCoordinates','SharpDXVectorsAndChartingCoordinates_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Understanding the SharpDX.Vector2**  SharpDX Draw methods use a [SharpDX.Vector2](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_vector2.htm) object which describes where to render a command relative to the chart panel.  These **Vector2** objects can be thought as a two-dimensional point in the chart panels X and Y axis. Since the chart canvas used to draw on consists of the full panel of the chart, a vector using a value of 0 for both the X and Y coordinates would  be located in the top left corner of the chart:     | ns | | --- | | // creates a vector located at the top left corner of the chart float x = 0; float y = 0; SharpDX.Vector2 myVector2 = new Vector2(x, y); |      |  | | --- | | **Tip**:   You can learn about [Understanding Chart Canvas Coordinates](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_chart_object_coordinates.htm) on another topic |     **Vector2** objects contain **X**and**Y** properties helpful to recalculate new properties based on the initial vector:     | ns | | --- | | float width = endPoint.X - startPoint.X;  float height = endPoint.Y - startPoint.Y; |     Additionally, you can recalculate a new vector from existing vector objects:     | ns | | --- | | SharpDX.Vector2 center = (startPoint + endPoint) / 2; |     It is also helpful to know that **Vector2** objects are similar to the [Windows Point](https://msdn.microsoft.com/en-us/library/system.windows.point(v=vs.110).aspx) structure and these two types can be used interchangeably.  Depending on the mechanism used to obtain user input or other application values, you may receive the coordinates in a **Point**.  For convenience, NinjaTrader provides a [DXExtension.ToVector2()](https://ninjatrader.com/es/support/helpGuides/nt8/dxextensions_tovector2.htm) method used for converting between these two objects if needed:     | ns | | --- | | SharpDX.Vector2 dxVector2 = wpfPoint.ToVector2(); |     **Calculating Chart Coordinates**  If you simply used a vector with static values, your **Vector2** objects would never change, and your drawing would remain fixed on a particular area of the chart (which may be desired).  However, since NinjaTrader charts are dynamic and responded to various market data updates, scroll, resize, and scale operations - you also need a way to recalculate **vectors** to display information dynamically. To assist in this process, NinjaTrader provides some GUI related utilities to help navigate the chart and calculate values for your custom rendering.     | ns | | --- | | // creates a vector located at the top left corner of the chart panel  startPoint = new SharpDX.Vector2(ChartPanel.X, ChartPanel.Y);    // creates a vector located at the bottom right corner of the chart panel  endPoint = new SharpDX.Vector2(ChartPanel.X + ChartPanel.W, ChartPanel.Y + ChartPanel.H); |     Common utilities fall under 4 key components, and you can learn more about their specific functions from the help guide topics linked in the table below:     |  |  | | --- | --- | | [ChartControl](https://ninjatrader.com/es/support/helpGuides/nt8/chartcontrol.htm) | The entire hosting grid of the Chart | | [ChartBars](https://ninjatrader.com/es/support/helpGuides/nt8/chartbars.htm) | The primary bars series configured on the Chart | | [ChartPanel](https://ninjatrader.com/es/support/helpGuides/nt8/chartpanel.htm) | The panel on which the calling script resides | | [ChartScale](https://ninjatrader.com/es/support/helpGuides/nt8/chartscale.htm) | The Y-Axis values of the configured ChartPanel |      |  | | --- | | **Note**:   For full absolute device coordinates always use **ChartPanel** X, Y, W, H values. **ChartScale** and **ChartControl** properties return WPF units, so they can be drastically different depending on DPI of the user's display.  You can learn about [Working with Pixel Coordinates](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_pixel_coordinates.htm) on another topic. | |

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

tog_minus        [SharpDX Brush Resources](javascript:HMToggle('toggle','SharpDXBrushResources','SharpDXBrushResources_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Understanding SharpDX Brush Resources**  To color or "paint" an area of the chart, you must define custom resources which describe how you wish the custom render to appear. **SharpDX** contains special resources modeled after the familiar [WPF Brushes](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_brushes.htm). However, the two objects are different in the way they are constructed and also in how they are managed after they are used.    There are many types of **SharpDX Brush Resources** which all derive from the same base [Direct2D1.Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) class.  This base object is not enough to describe how your object should be presented, so in order to use a brush for rendering purposes, you will need to determine exactly what type of brush you wish to use:     |  |  | | --- | --- | | [Direct2D1.SolidColorBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_solidcolorbrush.htm) | Paints an area with a solid color. | | [Direct2D1.RadialGradientBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_radialgradientbrush.htm) | Paints an area with a radial gradient. | | [Direct2D1.LinearGradientBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_lineargradientbrush.htm) | Paints an area with a linear gradient. |     **Describing SolidColorBrush Colors**  The most common and simple brush to use is a [Direct2D1.SolidColorBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_solidcolorbrush.htm)which allows you to paint using a solid color (or with transparency). In the most basic form, **SolidColorBrush** can be constructed using a predefined [SharpDX.Color](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_color.htm)     | ns | | --- | | SharpDX.Direct2D1.SolidColorBrush customDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.DodgerBlue); |     You can also use a [SharpDX.Color3](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_color3.htm) or [SharpDX.Color4](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_color4.htm) structure as a way to get more customizable colors in your rendering:     | ns | | --- | | // create a 3 component color using rgb values in float notation SharpDX.Color3 dxColor3 = new SharpDX.Color3(1.0f, 0.0f, 0.0f);   // create a 4 component color using rgb + alpha (transparency) in float notation SharpDX.Color4 dxColor4 = new SharpDX.Color4(dxColor3, 0.5f);   // solid color brush uses a Color4 during construction SharpDX.Direct2D1.SolidColorBrush argbColorBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, dxColor4); |     Alternatively, you can set the "transparency" of an existing brush by accessing its [Opacity](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush_opacity.htm) property:     | ns | | --- | | customDXBrush.Opacity = .25f; |      |  | | --- | | **Note**:  Unlike their [WPF counterparts](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_brushes.htm), **SharpDX** brushes are thread-safe and do **NOT** need to be frozen. |     **Converting SharpDX Brushes**  **SharpDX Brushes** are **device-dependent resources**, which means they can only be used with the device (i.e., [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm)) which created them.  In practice, this mean you should **ONLY** create your **SharpDX** brushes during the chart object's [OnRender()](https://ninjatrader.com/es/support/helpGuides/nt8/onrender.htm) or [OnRenderTargetChanged()](https://ninjatrader.com/es/support/helpGuides/nt8/onrendertargetchanged.htm) methods.     |  | | --- | | **Warning**:  Failure to create device-dependent resources during the **OnRender()** or **OnRenderTargetChanged()** can lead to a host of issues including memory and application corruption which can negatively impact the stability of NinjaTrader.  Please be careful your **SharpDX** device-dependent resources are only created and updated during either of these two run-time methods.  Please see the [Best Practices for SharpDX Resources](https://ninjatrader.com/es/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm#bestpracticesforsharpdxresources) section on this page for more information. |     Because of this detail, a common problem you may run into is the requirement to share a **SharpDX** device brush resource with a **WPF** application brush.  For example, you may have **WPF** brushes defined in the UI during [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) or recalculated conditionally during [OnBarUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onbarupdate.htm), but ultimately wish to use also in custom rendering routines.  For convenience, NinjaTrader provide a [DXExtension.ToDxBrush()](https://ninjatrader.com/es/support/helpGuides/nt8/dxextensions_todxbrush.htm) method used for converting these objects if necessary:     | ns | | --- | | areaBrushDx = areaBrush.ToDxBrush(RenderTarget);  smallAreaBrushDx = smallAreaBrush.ToDxBrush(RenderTarget);  textBrushDx = textBrush.ToDxBrush(RenderTarget); |      |  | | --- | | **Note**: If you are using a large number of brushes, and are not tied to WPF resources, you should favor creating the **SharpDX Brush** directly since the ToDxBrush() method can lead to performance issues if called too frequently during a single render pass.  Please see the [Best Practices for SharpDX Resources](https://ninjatrader.com/es/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm#bestpracticesforsharpdxresources) section on this page for more information. | |

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

tog_minus        [SharpDX RenderTarget](javascript:HMToggle('toggle','SharpDXRenderTarget','SharpDXRenderTarget_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Understanding the RenderTarget**  A [SharpDX Render Target](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) is a general purpose object resource used for receiving and executing drawing commands.  When using a NinjaTrader chart object, a pre-constructed Chart [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/rendertarget.htm) object is available for you to use and ready to receive commands.  You can think of the **RenderTarget** as the device context you are using to render to (i.e. the Chart Panel).  While there is nothing special you need to do to setup this resource, it is important to understand some details regarding the **RenderTarget** to learn how it can be used.    The **RenderTarget** is primarily used for executing commands such as drawing shapes or text:     | ns | | --- | | **RenderTarget**.DrawLine(startPoint, endPoint, areaBrushDx) |     It is commonly used for creating various resources such as **Brushes** and other **SharpDX** objects:     | ns | | --- | | SharpDX.Direct2D1.SolidColorBrush customDXBrush = new SharpDX.Direct2D1.SolidColorBrush(**RenderTarget**, SharpDX.Color.DodgerBlue); |     It can also be used to set various properties to describe how the **RenderTarget** should render:     | ns | | --- | | RenderTarget.AntialiasMode   = SharpDX.Direct2D1.AntialiasMode.PerPrimitive; |     **Sequencing RenderTarget commands**  If the sequence in which objects render is essential to your custom rendering, you will need to be mindful of the order in which you call various **RenderTarget** members. For example, we can draw a second line which uses a different [AntialiasMode](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_antialiasmode.htm) and the renders each line in the order the render target received its commands:     | ns | | --- | | RenderTarget.AntialiasMode = SharpDX.Direct2D1.AntialiasMode.Aliased; RenderTarget.DrawLine(startPoint, endPoint, areaBrushDx, 8);  RenderTarget.AntialiasMode = SharpDX.Direct2D1.AntialiasMode.PerPrimitive; RenderTarget.DrawLine(startPoint, endPoint, customDXBrush, 2); |     In the above example, this order of operations would result in the second [RenderTarget.DrawLine()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawline.htm) to be rendered "on top" of the first **RenderTarget.DrawLine().** If you instead called these two methods in reverse order, you would not see the thinner line since it would be covered up by the thicker line.     |  | | --- | | **Note**:  It is important to realize that **RenderTarget** **sequencing** and the [Chart Object ZOrder](https://ninjatrader.com/es/support/helpGuides/nt8/chart_zorder.htm) are two different concepts. The **ZOrder** property controls the overall layer your entire chart object appears relative to other chart objects existing on the same chart. **RenderTarget sequencing** only affects the order objects are rendered relative itself.  Therefore, it is not possible to sequence your chart object's **RenderTarget** to draw on two different **ZOrders** (e.g., one line above chart bars and another line below). |     **Using the RenderTarget with Device Resources**  Throughout the lifetime of a chart, the render target is created and destroyed several times to satisfy various user commands. As a result, any resources that are created need to be recreated and destroyed as that render target is updated.   The NinjaTrader [OnRenderTargetChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onrendertargetchanged.htm) method was designed to help with this process and will be called anytime the **RenderTarget** has changed.  You should use this method if you have objects which are passed around from various other resources.     |  | | --- | | **Warning**:  Failure to create device-dependent resources during the **OnRender()** or **OnRenderTargetChanged()** can lead to a host of issues including memory and application corruption which can negatively impact the stability of NinjaTrader.  Please be careful your **SharpDX** device-dependent resources are only created and updated during either of these two run-time methods.  Please see the [Best Practices for SharpDX Resources](https://ninjatrader.com/es/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm#bestpracticesforsharpdxresources) section on this page for more information. | |

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

tog_minus        [SharpDX Lines and Shapes](javascript:HMToggle('toggle','SharpDXLinesAndShapes','SharpDXLinesAndShapes_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **RenderTarget Draw Methods**  All drawings consistent of a few basic shapes which can be called through a handful of **RenderTarget** commands.  "Draw..." methods create just the outline of the shape, and "Fill..." will paint the interior of the shape.     |  |  | | --- | --- | | [RenderTarget.DrawEllipse()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawellipse.htm) | Draws the outline of the specified ellipse using the specified stroke style. | | [RenderTarget.DrawGeometry()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawgeometry.htm) | Draws the outline of the specified geometry using the specified stroke style. | | [RenderTarget.DrawLine()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawline.htm) | Draws a line between the specified points. | | [RenderTarget.DrawRectangle()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawrectangle.htm) | Draws the outline of a rectangle that has the specified dimensions and stroke style. | | [RenderTarget.FillEllipse()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillellipse.htm) | Paints the interior of the specified ellipse. | | [RenderTarget.FillGeometry()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillgeometry.htm) | Paints the interior of the specified geometry. | | [RenderTarget.FillRectangle()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillrectangle.htm) | Paints the interior of the specified rectangle. |      |  | | --- | | **Note**: [AntialiasMode.PerPrimitive](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_antialiasmode.htm) allows for graphics to render more sharply, but comes at a performance cost.  It is recommended to set the [RenderTarget.AntialiasMode](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_antialiasmode.htm) back to the default **AntialiasMode.Aliased** after you finish your **RenderTarget**Draw command.   Please see the [Best Practices for SharpDX Resources](https://ninjatrader.com/es/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm#bestpracticesforsharpdxresources) section on this page for more information. |     **Line**  The simplest shape is a Line, executed by the [RenderTarget.DrawLine()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawline.htm) command which just takes two [Vector2](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_vector2.htm) objects which describe where to draw the line, and (optionally) the width of the line to draw:     | ns | | --- | | // create two vectors for the line to draw  SharpDX.Vector2 startPoint = new SharpDX.Vector2(ChartPanel.X, ChartPanel.Y); SharpDX.Vector2 endPoint = new SharpDX.Vector2(ChartPanel.X + ChartPanel.W, ChartPanel.Y + ChartPanel.H);  // define the brush used in the line SharpDX.Direct2D1.SolidColorBrush customDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.DodgerBlue);  // execute the render target draw line with desired values RenderTarget.DrawLine(startPoint, endPoint, customDXBrush, 2);    // always dispose of a brush when finished  customDXBrush.Dispose(); | | render_target_drawline |     **Rectangle**  Using either the [RenderTarget.FillRectangle()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillrectangle.htm) or [RenderTarget.DrawRectangle()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawrectangle.htm) requires a [SharpDX.RectangleF](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_rectanglef.htm) structure, constructed using four values to represent the location (x, y) and size (width, height) of the rectangle to draw.     | ns | | --- | | // create two vectors to position the rectangle  SharpDX.Vector2 startPoint = new SharpDX.Vector2(ChartPanel.X, ChartPanel.Y); SharpDX.Vector2 endPoint = new SharpDX.Vector2(ChartPanel.X + ChartPanel.W, ChartPanel.Y + ChartPanel.H);  // calculate the desired width and heigh of the rectangle float width = endPoint.X - startPoint.X; float height = endPoint.Y - startPoint.Y;    // define the brush used in the rectangle  SharpDX.Direct2D1.SolidColorBrush customDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.DodgerBlue);    // construct the rectangleF struct to describe the with position and size the drawing SharpDX.RectangleF rect = new SharpDX.RectangleF(startPoint.X, startPoint.Y, width, height);  // execute the render target fill rectangle with desired values RenderTarget.FillRectangle(rect, customDXBrush);    // always dispose of a brush when finished  customDXBrush.Dispose(); | | render_target_drawrectangle |     **Ellipse**  Similar to the **Rectangle**, you can draw an **Ellipse** (or circle) using either the [RenderTarget.FillEllipse()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillellipse.htm) or [RenderTarget.DrawEllipse()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawellipse.htm) methods using a [SharpDX Direct2D1 Ellipse](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_ellipse.htm) struct.  For this structure, you will need to use a [Vector2](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_vector2.htm) object to determine the **Center** position of the ellipse, a **RadiusX,** and a **RadiusY** which determines the size of the ellipse:     | ns | | --- | | // create two vectors to position the ellipse  SharpDX.Vector2 startPoint = new SharpDX.Vector2(ChartPanel.X, ChartPanel.Y); SharpDX.Vector2 endPoint = new SharpDX.Vector2(ChartPanel.X + ChartPanel.W, ChartPanel.Y + ChartPanel.H);  // calculate the center point of the ellipse from start/end points  SharpDX.Vector2 centerPoint = (startPoint + endPoint) / 2;    // set the radius of the ellipse float radiusX = 50; float radiusY = 50;    // construct the rectangleF struct to describe the position and size the drawing  SharpDX.Direct2D1.Ellipse ellipse = new SharpDX.Direct2D1.Ellipse(centerPoint, radiusX, radiusY);    // define the brush used in the rectangle  SharpDX.Direct2D1.SolidColorBrush customDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.DodgerBlue);  // execute the render target fill ellipse with desired values RenderTarget.FillEllipse(ellipse, customDXBrush);    // always dispose of a brush when finished  customDXBrush.Dispose(); | | render_target_drawellipse |     **Geometry**  For more complicated shapes, you can use the [RenderTarget.FillGeometry()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillgeometry.htm) or [RenderTarget.DrawGeometry()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawgeometry.htm) methods using a [Direct2D1.PathGeometry](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_pathgeometry.htm) object, which is ultimately defined by a [Direct2D1.GeometrySink](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_geometrysink.htm) interface.     |  | | --- | | **Warning**:  Any **SharpDX PathGeometry** object used in your development must be disposed of after they have been used. NinjaTrader is **NOT** guaranteed to dispose of these resources for you!   Please see the [Best Practices for SharpDX Resources](https://ninjatrader.com/es/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm#bestpracticesforsharpdxresources) section on this page for more information. |     To describe a **PathGeometry** object's path, use the object's [PathGeometry.Open()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_pathgeometry_open.htm) method to retrieve an **GeometrySink**.  Then, use the **GeometrySink** to populate the geometry with figures and segments.  To create a figure, call the [GeometrySink.BeginFigure()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_geometrysink_beginfigure.htm) method, specify the figure's start point, and then use its Add methods (such as [GeometrySink.AddLine()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_geometrysink_addline.htm)) to add segments.  When you are finished adding segments, call the [GeometrySink.EndFigure()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_geometrysink_endfigure.htm) method. You can repeat this sequence to create additional figures. When you are finished creating figures, call the [GeometrySink.Close()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_geometrysink_close.htm) method.     | ns | | --- | | // create three vectors to position the geometry  SharpDX.Vector2 startPoint = new SharpDX.Vector2(ChartPanel.X, ChartPanel.Y); SharpDX.Vector2 endPoint = new SharpDX.Vector2(ChartPanel.X + ChartPanel.W, ChartPanel.Y + ChartPanel.H); SharpDX.Vector2 centerPoint = (startPoint + endPoint) / 2;   // create the PathGeometry used by the RenderTarget Fill/Draw method SharpDX.Direct2D1.PathGeometry trianglePathGeometry   = new SharpDX.Direct2D1.PathGeometry(Core.Globals.D2DFactory);   // retrieve the GeometrySink used to describe the PathGeometry SharpDX.Direct2D1.GeometrySink geometrySink   = trianglePathGeometry.Open();   // create the points used to define the GeometrySink SharpDX.Vector2 beginPoint = new SharpDX.Vector2(centerPoint.X, startPoint.Y);    // Create a figure using the beginPoint geometrySink.BeginFigure(beginPoint, SharpDX.Direct2D1.FigureBegin.Filled);  // add lines to the figure SharpDX.Vector2 line1 = new SharpDX.Vector2(endPoint.X, centerPoint.Y); geometrySink.AddLine(line1); SharpDX.Vector2 line2 = new SharpDX.Vector2(centerPoint.X, endPoint.Y); geometrySink.AddLine(line2);   // end and close figure when finished geometrySink.EndFigure(SharpDX.Direct2D1.FigureEnd.Closed); geometrySink.Close();  // define the brush used in the geometry  SharpDX.Direct2D1.SolidColorBrush customDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.DodgerBlue);    // execute the render target fill geometry with desired values RenderTarget.FillGeometry(trianglePathGeometry, customDXBrush);    // always dispose of a PathGeometry when finished  trianglePathGeometry.Dispose();  // always dispose of a brush when finished  customDXBrush.Dispose(); | | render_target_fillgeometry |      |  | | --- | | **Tip**:  For more examples of using **Shapes** for custom rendering, many of the DrawingTools included in the **NinjaTrader.Custom** project use these types of **SharpDX** objects and methods extensively. | |

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

tog_minus        [SharpDX Text Rendering](javascript:HMToggle('toggle','SharpDXTextRendering','SharpDXTextRendering_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Using SharpDX for rendering Text**  Up until this point, we have been using the [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) namespace to render shapes.  When dealing with text, there is a separate [SharpDX.DirectWrite](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite.htm) namespace which works along with the **Direct2D1** objects.    There are two principle objects used for text rendering:  A **TextFormat** object which sets the style of the text, and a **TextLayout** object used to construct complex texts with various settings and provides metrics for measuring the shape the formatted text.    Each one of these objects has their own **RenderTarget** methods: [RenderTarget.DrawText()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawtext.htm) for simple **TextFormat** objects and [RenderTarget.DrawTextLayout()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawtextlayout.htm) for more advanced layouts.  Both methods accept a **TextFormat** object; **DrawTextLayout** is more complicated but has better performance since it reuses the same text layout which does not need to be recalculated.     |  | | --- | | **Tip**:  Both the **TextFormat** and **TextLayout** objects require a **DirectWrite** factory during construction.  For convenience, you can simply use the pre-built NinjaTrader[.Core.Globals.DirectWriteFactory](https://ninjatrader.com/es/support/helpGuides/nt8/directwritefactory.htm) property. |     **Formatting Text**  The **TextFormat** object determines the font size, style and family, among other properties.     |  | | --- | | **Warning**:  Any **SharpDX TextFormat** object used in your development must be disposed of after they have been used. NinjaTrader is **NOT** guaranteed to dispose of these resources for you!  Please see the [Best Practices for SharpDX Resources](https://ninjatrader.com/es/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm#bestpracticesforsharpdxresources) section on this page for more information. |      | ns | | --- | | SharpDX.DirectWrite.TextFormat textFormat = new SharpDX.DirectWrite.TextFormat(Core.Globals.DirectWriteFactory, "Arial", 12); |     Once the text formatting has been described, you can use this object to immediately start rendering text in the DrawText() method.  This approach also requires a [SharpDX.RectangleF](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_rectanglef.htm) to help determine the size and position the text renders on the chart.     | ns | | --- | | // define the point for the text to render  SharpDX.Vector2 startPoint = new SharpDX.Vector2(ChartPanel.X, ChartPanel.Y);  // construct the text format with desired font family and size SharpDX.DirectWrite.TextFormat textFormat = new SharpDX.DirectWrite.TextFormat(Core.Globals.DirectWriteFactory, "Arial", 36);  // construct the rectangleF struct to describe the position and size the text SharpDX.RectangleF rectangleF = new SharpDX.RectangleF(startPoint.X, startPoint.Y, ChartPanel.W, ChartPanel.H);    // define the brush used for the text  SharpDX.Direct2D1.SolidColorBrush customDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.DodgerBlue);  // execute the render target text command with desired values RenderTarget.DrawText("I am some text", textFormat, rectangleF, customDXBrush);  // always dispose of textFormat when finished textFormat.Dispose();  // always dipose of brush when finished customDXBrush.Dispose(); | | render_target_drawtext |     **Converting Text**  One common approach to text formatting is to use the same formats as existing chart objects.  This provides familiar text format matching other objects which exist on the chart.  To accomplish this, you can simply use the **ChartControl** NinjaTrader[.Gui.SimpleFont](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm) object and convert to **SharpDX** using the [ToDirectWriteTextFormat()](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_todirectwritetextformat.htm) method.     | ns | | --- | | SharpDX.DirectWrite.TextFormat textFormat = ChartControl.Properties.LabelFont.ToDirectWriteTextFormat(); |     **Text Layouts**  The **TextLayout** object works in combination with the **TextFormat** object by extending its functionality and providing an interface more powerful than a simple Rectangle, enabling you to position, measure, or clip the text to a surrounding shape.    When constructing the **TextLayout** object, you will pass in the exact text as a string you wish to render, along with the desired **TextFormat**.  This gives you the ability to measure the text string after it has been formatted.  During construction, you also have an opportunity to specify the maximum height and width of the **TextLayout**.  For example, we can set the text layout to bound to height and width chart panel:     | ns | | --- | | SharpDX.DirectWrite.TextLayout textLayout = new SharpDX.DirectWrite.TextLayout(Core.Globals.DirectWriteFactory, "I am also some text", textFormat, ChartPanel.W, ChartPanel.H); |     After the text has its format and layout,  you can use the [RenderTarget.DrawTextLayout()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawtextlayout.htm) method to specify the exact location as a [Vector2](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_vector2.htm), as well as the [Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) used to draw the text.   | ns | | --- | | RenderTarget.DrawTextLayout(startPoint, textLayout, customDXBrush); |     **Measuring Text Layouts**  Working with an existing **TextLayout** object, you can use its [TextLayout.Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout_metrics.htm) object to retrieve metadata related to the size of the formatted text.   This is helpful if you are unsure of the size of the text before it is rendered.  For example, you may wish to draw a rectangle around the formatted text calculated width and height.  Using the approach below, the rectangle will dynamically resize to fit the text values used:     | ns | | --- | | // define the point for the text to render  SharpDX.Vector2 startPoint = new SharpDX.Vector2(ChartPanel.X + 20, ChartPanel.Y + 20);  // construct the text format with desired font family and size SharpDX.DirectWrite.TextFormat textFormat = new SharpDX.DirectWrite.TextFormat(Core.Globals.DirectWriteFactory, "Arial", 36);  // construct the text layout with desired text, text format, max width and height SharpDX.DirectWrite.TextLayout textLayout = new SharpDX.DirectWrite.TextLayout(Core.Globals.DirectWriteFactory, "I am also some text", textFormat, ChartPanel.W, ChartPanel.H);  // create a rectangle which will automatically resize to the width/height of the textLayout SharpDX.RectangleF rectangleF = new SharpDX.RectangleF(startPoint.X, startPoint.Y, textLayout.Metrics.Width, textLayout.Metrics.Height);    // define the brush used for the text and rectangle SharpDX.Direct2D1.SolidColorBrush customDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.DodgerBlue);   // execute the render target draw rectangle with desired values RenderTarget.DrawRectangle(rectangleF, customDXBrush);    // execute the render target text layout command with desired values RenderTarget.DrawTextLayout(startPoint, textLayout, customDXBrush);  // always dispose of textLayout, textFormat, or brush when finished textLayout.Dispose(); textFormat.Dispose(); customDXBrush.Dispose(); | | render_target_drawtextlayout |      |  | | --- | | **Note**:  The **TextLayout.Metrics** height and width properties return the text pixel height, including the line spacing of the font.  Due to the nature of most font families, there will be an amount of line spacing above and below the text.  You can use the [TextLayout.GetLineMetrics()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout_getlinemetrics.htm) method to help calculate the distance from the top of the text line to its baseline. | |

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

tog_minus        [SharpDX Stroke Style](javascript:HMToggle('toggle','SharpDXStrokeStyle','SharpDXStrokeStyle_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Using the StrokeStyle Object**  When rendering **SharpDX** Lines and Shapes, you can optionally configure a [SharpDX.Direct2D1.StrokeStyle](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle.htm) allowing you to utilize several pre-made [dash styles](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_dashstyle.htm), or even create a custom dash pattern.     |  | | --- | | **Note**:  Unlike other **SharpDX** objects such as **brushes**, the **StrokeStyle** is a device-independent resource.  This means you only need to create the object once throughout the lifetime of the script.  However, the **StrokeStyle** needs to be disposed of when the script is terminated.  The **Creating a Custom DashStyle** example below shows how to use a stroke style from the beginning to end of the lifetime of your script.   Please see the [Best Practices for SharpDX Resources](https://ninjatrader.com/es/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm#bestpracticesforsharpdxresources) section on this page for more information. |     For convenience, **SharpDX** provides the [StrokeStyleProperties](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyleproperties.htm) struct for creating new a **StrokeStyle:**     | ns | | --- | | // create a stroke style property using a pre-configured "DashDot" dash style SharpDX.Direct2D1.StrokeStyleProperties dxStrokeStyleProperties = new SharpDX.Direct2D1.StrokeStyleProperties {   DashStyle = SharpDX.Direct2D1.DashStyle.DashDot, }; |     Once you have your desired stroke style properties, you can create a new stroke style object.     |  | | --- | | **Warning**:  Any **SharpDX StrokeStyle** object used in your development must be disposed of after they have been used. NinjaTrader is **NOT** guaranteed to dispose of these resources for you!   Please see the [Best Practices for SharpDX Resources](https://ninjatrader.com/es/support/helpGuides/nt8/using_sharpdx_for_custom_chart_rendering.htm#bestpracticesforsharpdxresources) section on this page for more information. |      | ns | | --- | | SharpDX.Direct2D1.StrokeStyle dxStrokeStyle = new SharpDX.Direct2D1.StrokeStyle(NinjaTrader.Core.Globals.D2DFactory, dxStrokeStyleProperties); |      |  | | --- | | **Tip**:  The **SharpDX.Direct2D1.StrokeStyle** require a **Direct2D1** factory during construction.  For convenience, you can simply use the pre-built NinjaTrader[.Core.Globals.D2DFactory](https://ninjatrader.com/es/support/helpGuides/nt8/d2dfactory.htm) property. |     And then use that object with the RenderTarget.DrawLine() method:   | ns | | --- | | RenderTarget.DrawLine(startPoint, endPoint, dxBrush, width, dxStrokeStyle); |     **Creating a Custom DashStyle**  By setting the [StrokeStyle.DashStyle](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle_dashstyle.htm) property to "**Custom**", you can further refine the appearance of a **SharpDX** rendered line or shape by describing the length and space between the lines. Creating a custom **DashStyle** is not only useful for using **RenderTarget methods**, but also can be used for customizing the appearance of standard [NinjaScript Plots](https://ninjatrader.com/es/support/helpGuides/nt8/addplot.htm).    The code example creates a single **StrokeStyle** object using custom dash style properties.  The example then uses those the custom stroke style object with user defined dashes for overriding the default NinjaTrader plot appearances, and using the same stroke style in a [RenderTarget.DrawLine()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawline.htm) command.     | ns | | --- | | // a SharpDX.Direct2D1.StrokeStyle is device independent // it only needs to be setup once throughout the lifetime of your script private SharpDX.Direct2D1.StrokeStyle dxStrokeStyle;   protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "Custom StrokeStyle";       AddPlot(Brushes.Blue, "Custom StrokeStyle");   }   else if (State == State.Configure)   {     // create a custom stroke style when configured     SharpDX.Direct2D1.StrokeStyleProperties dxStrokeStyleProperties = new SharpDX.Direct2D1.StrokeStyleProperties     {         // set the dash style to "Custom" define the dash pattern         DashStyle = SharpDX.Direct2D1.DashStyle.Custom,           // set further custom/optional StrokeStyle appearances         DashCap = CapStyle.Round,         EndCap   = CapStyle.Flat,         StartCap = CapStyle.Square,         LineJoin = LineJoin.Miter,           // offset in the dash sequence         DashOffset = 10.0f,     };       // define the an array of floating-point values     float[] dashes = { 1.0f, 2.0f, 2.0f, 3.0f, 2.0f, 2.0f };       // create the stroke style using the custom properties and dash array     dxStrokeStyle = new SharpDX.Direct2D1.StrokeStyle(NinjaTrader.Core.Globals.D2DFactory,             dxStrokeStyleProperties, dashes);   }   else if (State == State.Terminated)   {     // make sure to dispose of stroke style when finished     if (dxStrokeStyle != null)     {         if (!dxStrokeStyle.IsDisposed)           dxStrokeStyle.Dispose();     }   } }   protected override void OnBarUpdate() {   Value[0] = Close[0]; }   protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // override the appearance of the default plot style   Plots[0].StrokeStyle = dxStrokeStyle;   base.OnRender(chartControl, chartScale);     // use the custom dash style in a RenderTarget.DrawLine() commands   using ( SharpDX.Direct2D1.SolidColorBrush dxBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.Blue))   {     RenderTarget.DrawLine(new SharpDX.Vector2(ChartPanel.X, ChartPanel.Y), new SharpDX.Vector2(ChartPanel.X + ChartPanel.W, ChartPanel.Y + ChartPanel.H), dxBrush, 2, dxStrokeStyle);   } } | | SharpDX_StrokeStyle | |

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

tog_minus        [Best Practices for SharpDX Resources](javascript:HMToggle('toggle','BestPracticesforSharpDXResources','BestPracticesforSharpDXResources_ICON'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Understanding Device-dependent vs Device-independent resources**  Direct2D has several types of resources which may be mapped to the different hardware devices:    •**Device-independent** resources are on the CPU  •**Device-dependent** resources are on the GPU    When **device-dependent** resources are created, system resources are dedicated to that object.  Resources which are **device-dependent** are associated with a particular **RenderTarget** device and are only available on that device.  Therefore, objects which were created using a **RenderTarget** can only be used by that device.  As the **RenderTarget** updates, objects which were previously created will no longer be compatible and can lead to errors.  You can use the NinjaTrader [OnRenderTargetChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onrendertargetchanged.htm)method to detect when the render target has updated and gives you an opportunity to recreate resources.    **Device-dependent resources**  The following objects are associated with a specific **RenderTarget**.  They must be created and dispose of any time the **RenderTarget** is updated:    •[Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm)  •[GeometrySink](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_geometrysink.htm)  •[GradientStopCollection](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_gradientstopcollection.htm)  •[LinearGradientBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_lineargradientbrush.htm)  •[RadialGradientBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_radialgradientbrush.htm)  •[SolidColorBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_solidcolorbrush.htm)    **Device-independent resources**  The following objects are **NOT** associated with a specific device.  They can be created once and last for the lifetime of your script, or until they need to be modified:    •[PathGeometry](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_pathgeometry.htm)  •[StrokeStyle](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle.htm)  •[TextFormat](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textformat.htm)  •[TextLayout](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout.htm)     |  | | --- | | **Note**:  For more technical information on device resources, please see the [MSDN Direct2D Resources Overview](https://msdn.microsoft.com/en-us/library/dd756757(v=vs.85).aspx) |     **SharpDX DisposeBase**  Although most C# objects stored in memory are handled by the operating system, there are a few **SharpDX** resources which are not managed.  It is important to take care of these resources during the lifetime of your script as there is no guarantee that NinjaTrader will be able to dispose of these unmanaged references for you.    The following commonly used objects implement from the [SharpDX.DisposeBase](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase.htm) and should be disposed any time they are created:    •[Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm)  •[GeometrySink](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_geometrysink.htm)  •[GradientStopCollection](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_gradientstopcollection.htm)  •[LinearGradientBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_lineargradientbrush.htm)  •[PathGeometry](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_pathgeometry.htm)  •[RadialGradientBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_radialgradientbrush.htm)  •[SolidColorBrush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_solidcolorbrush.htm)  •[StrokeStyle](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle.htm)  •[TextFormat](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textformat.htm)  •[TextLayout](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout.htm)     |  | | --- | | **Warning**:  The list above is **NOT** exhaustive and there are other less common **SharpDX** objects that could implement **DisposeBase**. Failure to clean up these resources **WILL** result in NinjaTrader using more memory than necessary and may expose potential "memory leaks" coming from your script.  If you experience unusual amounts of memory being utilized over time, an unmanaged **SharpDX** resource is often times the culprit. |     Since there is no guarantee that NinjaTrader will release objects from memory when your script is terminated, it is best to protect these resources from issues and call [Dispose()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase_dispose.htm) as soon as possible.  This commonly involves calling **Dispose(**) at the end of [OnRender()](https://ninjatrader.com/es/support/helpGuides/nt8/onrender.htm),or during [OnRenderTargetChanged()](https://ninjatrader.com/es/support/helpGuides/nt8/onrendertargetchanged.htm) when dealing with **device- dependent** resources such as brush. **Device-independent** resources can be created once and then retained for the life of your application.     | ns | | --- | | protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   // 1 - setup your resource   SharpDX.Direct2D1.SolidColorBrush customDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.DodgerBlue   // 2 - use your resource   RenderTarget.DrawLine(startPoint, endPoint, customDXBrush);         // 3- dispose of your resource   customDXBrush.Dispose() } |      |  | | --- | | **Note**:  If your resource is setup (i.e., uses the "new" keyword) during **OnRender()** or **OnRenderTargetChange()**, calling **.Dispose()** during [State.Terminated](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm) will **ONLY** dispose of the *very last reference in memory* and is **NOT** sufficient to completely manage all instances created during the lifetime of your script.  You should be diligent in calling **Dispose()**throughout the lifetime of the script. |     You can also consider implementing the [using Statement (C# Reference)](https://msdn.microsoft.com/en-us/library/yh598w02.aspx) which will implicitly call **Dispose() for** you when you are done:     | ns | | --- | | // customDXBrush implicitly calls Dispose() after this block executes  using (SharpDX.Direct2D1.SolidColorBrush customDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.DodgerBlue)) {   RenderTarget.DrawLine(startPoint, endPoint, customDXBrush); } |      |  | | --- | | **Critical**:  Attempting to use an object which has already been disposed can lead to memory corruption that NinjaTrader may not be able to recover.  Attempts to use an object in this manner can result in an error similar to: **Error on calling 'OnRender' method on bar 0: Attempted to read or write protected memory. This is often an indication that other memory is corrupt.** |       You can check to see if can object has been disposed of by using the [DisposeBase.IsDiposed](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase_isdisposed.htm) property:     | ns | | --- | | SharpDX.Direct2D1.Brush customDXBrush = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.DodgerBlue);  // checks the object is not disposed of before using if(!customDXBrush.IsDisposed) {   RenderTarget.DrawLine(startPoint, endPoint, customDXBrush);   customDXBrush.Dispose(); } |     You should also favor managing these resources yourself, which means methods which accept a **SharpDX DisposeBase** object as an argument should be created before they are passed into the method and disposed of after they are used.  For example, the code below should be avoided:     | ns | **Practice to avoid** | | --- | --- | | // do NOT convert an object as it is passed to an argument.  // You may have no chance to Dispose of the object!  // Finalizer is not guaranteed to release of these resources RenderTarget.DrawLine(startPoint, endPoint, Brushes.AliceBlue.ToDxBrush(RenderTarget));    MyCustomMethod(Brushes.AliceBlue.ToDxBrush(RenderTarget)); | |     Instead, you should manage these objects yourself:   | ns **Best practice** | | --- | | // Do create and store this reference yourself so you can control when it is released (Y) SharpDX.Direct2D1.Brush customDXBrush = WPFBrush.ToDxBrush(RenderTarget);    RenderTarget.DrawLine(startPoint, endPoint, customDXBrush));    MyCustomMethod(customDXBrush);    customDXBrush.Dipose() |     **Other Best Practices**    If possible, you should avoid using the [ToDxBrush()](https://ninjatrader.com/es/support/helpGuides/nt8/dxextensions_todxbrush.htm) method if it is not necessary.  It is relatively harmless to use this approach for a few brushes, but can introduce performance issues if used too liberally.     | ns **Practice to avoid** | | --- | | // do NOT convert from WPF brushes unnecessarily  SharpDX.Direct2D1.Brush dxBrush1 = System.Windows.Media.Brushes.Blue.ToDxBrush(RenderTarget); SharpDX.Direct2D1.Brush dxBrush2 = System.Windows.Media.Brushes.Red.ToDxBrush(RenderTarget); SharpDX.Direct2D1.Brush dxBrush3 = System.Windows.Media.Brushes.Green.ToDxBrush(RenderTarget); SharpDX.Direct2D1.Brush dxBrush4 = System.Windows.Media.Brushes.Purple.ToDxBrush(RenderTarget); SharpDX.Direct2D1.Brush dxBrush5 = System.Windows.Media.Brushes.Orange.ToDxBrush(RenderTarget); SharpDX.Direct2D1.Brush dxBrush6 = System.Windows.Media.Brushes.Yellow.ToDxBrush(RenderTarget); |     Instead, you should construct a SharpDX Brush directly if a WPF brush is not ever needed:   | ns **Best practice** | | --- | | // Do create SharpDX Brushes directly if you have a large amount of brushes SharpDX.Direct2D1.Brush dxBrush1 = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.Blue); SharpDX.Direct2D1.Brush dxBrush2 = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.Red); SharpDX.Direct2D1.Brush dxBrush3 = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.Green); SharpDX.Direct2D1.Brush dxBrush4 = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.Purple); SharpDX.Direct2D1.Brush dxBrush5 = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.Orange); SharpDX.Direct2D1.Brush dxBrush6 = new SharpDX.Direct2D1.SolidColorBrush(RenderTarget, SharpDX.Color.Yellow); |     Rendering with anti-aliasing disabled can be used to render a higher qualify shapes but comes as a performance impact.  You should make sure to set this render target property back to its default when you are finished with a render routine.     | ns **Best practice** | | --- | | // AntialiasMode.PerPrimitive is more resource intensive  // store the old reference before setting the desired value SharpDX.Direct2D1.AntialiasMode oldAntialiasMode = RenderTarget.AntialiasMode; RenderTarget.AntialiasMode = SharpDX.Direct2D1.AntialiasMode.PerPrimitive;   // execute your render routines   // and then set back to initial AntialiasMode when finished RenderTarget.AntialiasMode = oldAntialiasMode; | |

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

|  |  |
| --- | --- |
| **Navigation:**  »No topics above this level«  **8.0.0.13 (RC1) Release Notes** | [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout.htm) |

**Release Date**

August 31st, 2016

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

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

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

**Code Breaking Changes**

**Compile Errors**

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

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

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

**Implementation changes**

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

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

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

**Notes**

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

**Definition**

Indicates if the chart anchor was drawn by a NinjaScript object (such as an indicator or strategy).

**Property Value**

A bool value which returns **true** of the object was drawn by other NinjaScript object; otherwise **false**.  This property is read-only.

**Syntax**

<ChartAnchor>.IsNinjaScriptDrawn

**Examples**

| ns | |
| --- | --- |
| //unlocks the NinjaScript drawn object and allows the user to modify the anchor, while the NinjaScript object still 'owns' the object protected override void OnBarUpdate() {     foreach(IDrawingTool dt in DrawObjects)         {           DrawingTools.Line sampleLine = dt as DrawingTools.Line;                       if (sampleLine != null && sampleLine.StartAnchor.IsNinjaScriptDrawn)           {               sampleLine.IsLocked = false;               Print(sampleLine.StartAnchor.ToString());           }         } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) >  **SharpDX.Direct2D1.RenderTarget** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_radialgradientbrushproperties.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_antialiasmode.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | | |

**Definition**

Represents an object that can receive drawing commands.

(See also [unmanaged API documentation](http://msdn.microsoft.com/en-us/library/dd371766.aspx))

**Syntax**

class RenderTarget

|  |
| --- |
| **Tips**:  1.For NinjaScript Development purposes, [DrawingTools](https://ninjatrader.com/es/support/helpGuides/nt8/drawingtool.htm), [ChartStyles](https://ninjatrader.com/es/support/helpGuides/nt8/chartstyletype.htm), [Indicators](https://ninjatrader.com/es/support/helpGuides/nt8/indicators.htm), and [Strategies](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) implement the Chart's [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/rendertarget.htm) ready to be used in the **OnRender()** method  2.General information on **Direct2D** **Render Targets** can be found on the [MSDN Direct2D Render Targets Overview](https://msdn.microsoft.com/en-us/library/dd756757(v=vs.85).aspx) |

**Methods and Properties**

|  |  |
| --- | --- |
| [AntialiasMode](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_antialiasmode.htm) | Retrieves or sets the current antialiasing mode for nontext drawing operations. |
| [DrawEllipse()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawellipse.htm) | Draws the outline of the specified ellipse using the specified stroke style. |
| [DrawGeometry()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawgeometry.htm) | Draws the outline of the specified geometry. |
| [DrawLine()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawline.htm) | Draws a line between the specified points. |
| [DrawRectangle()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawrectangle.htm) | Draws the outline of a rectangle that has the specified dimensions. |
| [DrawText()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawtext.htm) | Draws the specified text using the format information provided by an [SharpDX.DirectWrite.TextFormat](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textformat.htm) object. |
| [DrawTextLayout()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawtextlayout.htm) | Draws the formatted text described by the specified [SharpDX.DirectWrite.TextLayout](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout.htm) object. |
| [FillEllipse()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillellipse.htm) | Paints the interior of the specified ellipse. |
| [FillGeometry()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillgeometry.htm) | Paints the interior of the specified geometry. |
| [FillRectangle()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillrectangle.htm) | Paints the interior of the specified rectangle. |
| [IsDisposed](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase_isdisposed.htm) | Gets a value indicating whether this instance is disposed.  (Inherited from [SharpDX.DisposeBase](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase.htm).) |
| [Transform](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_transform.htm) | Gets or sets the current transform of the render target. |

[Ir al contenido principal](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_antialias_mode?redirectedfrom=MSDN#main)

[**Learn**](https://learn.microsoft.com/es-mx/)

* Detectar
* Documentación del producto
* Lenguajes de desarrollo
* Temas

[Iniciar sesión](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_antialias_mode?redirectedfrom=MSDN)

[**Desarrollo de aplicaciones de Windows**](https://learn.microsoft.com/es-mx/windows/apps)

* Explorar
* Desarrollo
* Plataformas
* [Solución de problemas](https://learn.microsoft.com/es-mx/troubleshoot/windows/win32/)
* Recursos

[**Panel**](https://partner.microsoft.com/dashboard)

 Es posible que algunas partes de este tema se tradujeran mediante traducción automática o IA.

Ignorar alerta

Principio del formulario

Buscar

Final del formulario

* [Direct2D](https://learn.microsoft.com/es-mx/windows/win32/api/_direct2d/)
* D2d1.h
  + [Información general](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/)
  + [**enumeración D2D1\_ANTIALIAS\_MODE**](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_antialias_mode)
  + [estructura de D2D1\_ARC\_SEGMENT](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_arc_segment)
  + [enumeración D2D1\_ARC\_SIZE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_arc_size)
  + [estructura de D2D1\_BEZIER\_SEGMENT](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_bezier_segment)
  + [estructura de D2D1\_BITMAP\_BRUSH\_PROPERTIES](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_bitmap_brush_properties)
  + [enumeración D2D1\_BITMAP\_INTERPOLATION\_MODE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_bitmap_interpolation_mode)
  + [estructura de D2D1\_BITMAP\_PROPERTIES](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_bitmap_properties)
  + [estructura de D2D1\_BRUSH\_PROPERTIES](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_brush_properties)
  + [enumeración D2D1\_CAP\_STYLE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_cap_style)
  + [enumeración D2D1\_COMBINE\_MODE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_combine_mode)
  + [enumeración D2D1\_COMPATIBLE\_RENDER\_TARGET\_OPTIONS](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_compatible_render_target_options)
  + [enumeración D2D1\_DASH\_STYLE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_dash_style)
  + [enumeración D2D1\_DC\_INITIALIZE\_MODE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_dc_initialize_mode)
  + [enumeración D2D1\_DEBUG\_LEVEL](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_debug_level)
  + [enumeración D2D1\_DRAW\_TEXT\_OPTIONS](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_draw_text_options)
  + [estructura de D2D1\_DRAWING\_STATE\_DESCRIPTION](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_drawing_state_description)
  + [estructura de D2D1\_ELLIPSE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_ellipse)
  + [enumeración D2D1\_EXTEND\_MODE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_extend_mode)
  + [estructura de D2D1\_FACTORY\_OPTIONS](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_factory_options)
  + [enumeración D2D1\_FACTORY\_TYPE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_factory_type)
  + [enumeración D2D1\_FEATURE\_LEVEL](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_feature_level)
  + [enumeración D2D1\_FIGURE\_BEGIN](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_figure_begin)
  + [enumeración D2D1\_FIGURE\_END](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_figure_end)
  + [enumeración D2D1\_FILL\_MODE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_fill_mode)
  + [enumeración D2D1\_GAMMA](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_gamma)
  + [enumeración D2D1\_GEOMETRY\_RELATION](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_geometry_relation)
  + [enumeración D2D1\_GEOMETRY\_SIMPLIFICATION\_OPTION](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_geometry_simplification_option)
  + [estructura de D2D1\_GRADIENT\_STOP](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_gradient_stop)
  + [estructura de D2D1\_HWND\_RENDER\_TARGET\_PROPERTIES](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_hwnd_render_target_properties)
  + [enumeración D2D1\_LAYER\_OPTIONS](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_layer_options)
  + [estructura de D2D1\_LAYER\_PARAMETERS](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_layer_parameters)
  + [enumeración D2D1\_LINE\_JOIN](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_line_join)
  + [estructura de D2D1\_LINEAR\_GRADIENT\_BRUSH\_PROPERTIES](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_linear_gradient_brush_properties)
  + [enumeración D2D1\_OPACITY\_MASK\_CONTENT](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_opacity_mask_content)
  + [enumeración D2D1\_PATH\_SEGMENT](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_path_segment)
  + [enumeración D2D1\_PRESENT\_OPTIONS](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_present_options)
  + [estructura de D2D1\_QUADRATIC\_BEZIER\_SEGMENT](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_quadratic_bezier_segment)
  + [estructura de D2D1\_RADIAL\_GRADIENT\_BRUSH\_PROPERTIES](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_radial_gradient_brush_properties)
  + [estructura de D2D1\_RENDER\_TARGET\_PROPERTIES](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_render_target_properties)
  + [enumeración D2D1\_RENDER\_TARGET\_TYPE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_render_target_type)
  + [enumeración D2D1\_RENDER\_TARGET\_USAGE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_render_target_usage)
  + [estructura de D2D1\_ROUNDED\_RECT](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_rounded_rect)
  + [estructura de D2D1\_STROKE\_STYLE\_PROPERTIES](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_stroke_style_properties)
  + [enumeración D2D1\_SWEEP\_DIRECTION](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_sweep_direction)
  + [enumeración D2D1\_TEXT\_ANTIALIAS\_MODE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_text_antialias_mode)
  + [estructura de D2D1\_TRIANGLE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ns-d2d1-d2d1_triangle)
  + [enumeración D2D1\_WINDOW\_STATE](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_window_state)
  + [Función D2D1CreateFactory](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/nf-d2d1-d2d1createfactory)
  + [Función D2D1CreateFactory](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/nf-d2d1-d2d1createfactory-r1)
  + [Función D2D1InvertMatrix](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/nf-d2d1-d2d1invertmatrix)
  + [Función D2D1IsMatrixInvertible](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/nf-d2d1-d2d1ismatrixinvertible)
  + [Función D2D1MakeRotateMatrix](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/nf-d2d1-d2d1makerotatematrix)
  + [Función D2D1MakeSkewMatrix](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/nf-d2d1-d2d1makeskewmatrix)
  + Interfaz ID2D1Bitmap
  + Interfaz ID2D1BitmapBrush
  + Interfaz ID2D1BitmapRenderTarget
  + Interfaz ID2D1Brush
  + Interfaz ID2D1DCRenderTarget
  + Interfaz ID2D1DrawingStateBlock
  + Interfaz ID2D1EllipseGeometry
  + Interfaz ID2D1Factory
  + Interfaz ID2D1GdiInteropRenderTarget
  + Interfaz ID2D1Geometry
  + Interfaz ID2D1GeometryGroup
  + Interfaz ID2D1GeometrySink
  + Interfaz ID2D1GradientStopCollection
  + Interfaz ID2D1HwndRenderTarget
  + [Interfaz ID2D1Image](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/nn-d2d1-id2d1image)
  + Interfaz ID2D1Layer
  + Interfaz ID2D1LinearGradientBrush
  + Interfaz ID2D1Mesh
  + Interfaz ID2D1PathGeometry
  + Interfaz ID2D1RadialGradientBrush
  + Interfaz ID2D1RectangleGeometry
  + Interfaz ID2D1RenderTarget
  + Interfaz ID2D1Resource
  + Interfaz ID2D1RoundedRectangleGeometry
  + Interfaz ID2D1SimplifiedGeometrySink
  + Interfaz ID2D1SolidColorBrush
  + Interfaz ID2D1StrokeStyle
  + Interfaz ID2D1TessellationSink
  + Interfaz ID2D1TransformedGeometry
* D2d1\_1.h
* D2d1\_1helper.h
* D2d1\_2.h
* D2d1\_3.h
* D2d1\_3helper.h
* D2d1effectauthor.h
* D2d1effectauthor\_1.h
* D2d1effecthelpers.h
* D2d1effecthelpers.h
* D2d1effects.h
* D2d1effects\_1.h
* D2d1effects\_2.h
* D2d1helper.h
* D2d1svg.h
* Dcommon.h
* Winerror.h

Descargar PDF

1. [Learn](https://learn.microsoft.com/es-mx/)

1. [Windows](https://learn.microsoft.com/es-mx/windows/)

1. [Aplicaciones](https://learn.microsoft.com/es-mx/windows/apps/)

1. [Win32](https://learn.microsoft.com/es-mx/windows/win32/)

1. [API](https://learn.microsoft.com/es-mx/windows/win32/api/)

1. [Direct2D](https://learn.microsoft.com/es-mx/windows/win32/api/_direct2d/)

1. [D2d1.h](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/)

[**Leer en inglés**](https://learn.microsoft.com/en-us/windows/win32/api/d2d1/ne-d2d1-d2d1_antialias_mode?redirectedfrom=MSDN)Guardar

**enumeración D2D1\_ANTIALIAS\_MODE (d2d1.h)**

* Artículo
* 24/02/2024

Comentarios

**En este artículo**

1. [Syntax](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_antialias_mode?redirectedfrom=MSDN#syntax)
2. [Constantes](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_antialias_mode?redirectedfrom=MSDN#constants)
3. [Requisitos](https://learn.microsoft.com/es-mx/windows/win32/api/d2d1/ne-d2d1-d2d1_antialias_mode?redirectedfrom=MSDN#requirements)

Especifica cómo se representan los bordes de primitivos que no son de texto.

**Syntax**

C++Copiar

typedef enum D2D1\_ANTIALIAS\_MODE {

D2D1\_ANTIALIAS\_MODE\_PER\_PRIMITIVE = 0,

D2D1\_ANTIALIAS\_MODE\_ALIASED = 1,

D2D1\_ANTIALIAS\_MODE\_FORCE\_DWORD = 0xffffffff

} ;

**Constantes**

Expandir tabla

|  |
| --- |
| D2D1\_ANTIALIAS\_MODE\_PER\_PRIMITIVE Valor: *0* Los bordes están suavizados mediante el método Direct2D por primitivo de suavizado de contorno de alta calidad. |
| D2D1\_ANTIALIAS\_MODE\_ALIASED Valor: *1* Los objetos se aliasan en la mayoría de los casos. Los objetos solo se suavizan cuando se dibujan en un destino de representación creado por el método [CreateDxgiSurfaceRenderTarget](https://learn.microsoft.com/es-es/windows/win32/api/d2d1/nf-d2d1-id2d1factory-createdxgisurfacerendertarget(idxgisurface_constd2d1_render_target_properties__id2d1rendertarget)) y el muestreo múltiple de Direct3D se ha habilitado en la superficie de respaldo de DirectX Graphics Infrastructure (DXGI). |
| D2D1\_ANTIALIAS\_MODE\_FORCE\_DWORD Valor: *0xffffffff* |

**Requisitos**

Expandir tabla

| **Requisito** | **Value** |
| --- | --- |
| **Cliente mínimo compatible** | Windows 7, Windows Vista con SP2 y Platform Update para Windows Vista [aplicaciones de escritorio | Aplicaciones para UWP] |
| **Servidor mínimo compatible** | Windows Server 2008 R2, Windows Server 2008 con SP2 y Actualización de plataforma para Windows Server 2008 [aplicaciones de escritorio | Aplicaciones para UWP] |
| **Encabezado** | d2d1.h |

**Comentarios**

**¿Le resultó útil esta página?**

SíNo

[Enviar comentarios del producto](https://www.microsoft.com/en-us/windowsinsider/feedbackhub/fb)|

[Obtén ayuda en Microsoft Q&A](https://learn.microsoft.com/answers/tags/224/windows-api-win32/)

[**Español (México)**](https://learn.microsoft.com/es-mx/locale?target=https%3A%2F%2Flearn.microsoft.com%2Fes-mx%2Fwindows%2Fwin32%2Fapi%2Fd2d1%2Fne-d2d1-d2d1_antialias_mode%3Fredirectedfrom%3DMSDN)

[**Sus opciones de privacidad**](https://aka.ms/yourcaliforniaprivacychoices)

Tema

* [Versiones anteriores](https://learn.microsoft.com/es-mx/previous-versions/)
* [Blog](https://techcommunity.microsoft.com/t5/microsoft-learn-blog/bg-p/MicrosoftLearnBlog)
* [Colaborar](https://learn.microsoft.com/es-mx/contribute/)
* [Privacidad](https://go.microsoft.com/fwlink/?LinkId=521839)
* [Términos de uso](https://learn.microsoft.com/es-mx/legal/termsofuse)
* [Marcas comerciales](https://www.microsoft.com/legal/intellectualproperty/Trademarks/)
* © Microsoft 2025

|  |  |
| --- | --- |
| * **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) > [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) > * **SharpDX.Direct2D1.RenderTarget.DrawGeometry()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawellipse.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawline.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | |

## Definition

Draws the outline of the specified geometry using the specified stroke style.

(See also [unmanaged API documentation](http://msdn.microsoft.com/en-us/library/dd371890.aspx))

|  |
| --- |
| **Note**:  This method doesn't return an error code if it fails. |

## Method Return Value

This method does not return a value

## Syntax

RenderTarget.DrawGeometry(Geometry geometry, Brush brush)  
RenderTarget.DrawGeometry(Geometry geometry, Brush brush, float strokeWidth)  
RenderTarget.DrawGeometry(Geometry geometry, Brush brush, float strokeWidth, StrokeStyle strokeStyle)

## Parameters

|  |  |
| --- | --- |
| brush | An int which represents the method input |
| geometry | The [SharpDX.Direct2D1.Geometry](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_pathgeometry.htm) to draw |
| strokeStyle | The [SharpDX.Direct2D1.StrokeStyle](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle.htm) to apply to the geometry's outline, or null to paint a solid stroke. |
| strokeWidth | The thickness of the geometry's stroke. The stroke is centered on the geometry's outline. |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) > [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) >  **SharpDX.Direct2D1.RenderTarget.DrawLine()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawgeometry.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawrectangle.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | | |

**Definition**

Draws a line between the specified points.

(See also [unmanaged API documentation](http://msdn.microsoft.com/en-us/library/dd371895.aspx))

|  |
| --- |
| **Note**:  This method doesn't return an error code if it fails. |

**Method Return Value**

This method does not return a value

**Syntax**

RenderTarget.DrawLine(Vector2 point0, Vector2 point1, Brush brush)  
RenderTarget.DrawLine(Vector2 point0, Vector2 point1, Brush brush, float strokeWidth)  
RenderTarget.DrawLine(Vector2 point0, Vector2 point1, Brush brush, float strokeWidth, StrokeStyle strokeStyle)

**Parameters**

|  |  |
| --- | --- |
| brush | The [SharpDX.Direct2D1.Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) brush used to paint the line's stroke. |
| point0 | A [SharpDX.Vector2](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_vector2.htm) which determines the start point of the line, in device-independent pixels. |
| point1 | A [SharpDX.Vector2](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_vector2.htm) which determines the end point of the line, in device-independent pixels. |
| strokeStyle | The [SharpDX.Direct2D1.StrokeStyle](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle.htm) to paint, or null to paint a solid line. |
| strokeWidth | A value greater than or equal to 0.0f that specifies the width of the stroke. If this parameter isn't specified, it defaults to 1.0f. The stroke is centered on the line. |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) > [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) >  **SharpDX.Direct2D1.RenderTarget.DrawRectangle()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawline.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawtext.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | | |

**Definition**

Draws the outline of a rectangle that has the specified dimensions and stroke style.

(See also [unmanaged API documentation](http://msdn.microsoft.com/en-us/library/dd371902.aspx))

|  |
| --- |
| **Note**:  This method doesn't return an error code if it fails. |

**Method Return Value**

This method does not return a value

**Syntax**

RenderTarget.DrawRectangle(RectangleF rect, Brush brush)  
RenderTarget.DrawRectangle(RectangleF rect, Brush brush, float strokeWidth)  
RenderTarget.DrawRectangle(RectangleF rect, Brush brush, float strokeWidth, StrokeStyle strokeStyle)

**Parameters**

|  |  |
| --- | --- |
| brush | The [SharpDX.Direct2D1.Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) used to paint the rectangle's stroke. |
| rect | The [SharpDX.RectangleF](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_rectanglef.htm) which determines the dimensions of the rectangle to draw, in device-independent pixels. |
| strokeStyle | The [SharpDX.Direct2D1.StrokeStyle](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_strokestyle.htm) used to paint, or null to paint a solid stroke. |
| strokeWidth | A value greater than or equal to 0.0f that specifies the width of the rectangle's stroke. The stroke is centered on the rectangle's outline. |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) > [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) >  **SharpDX.Direct2D1.RenderTarget.DrawText()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawrectangle.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawtextlayout.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | | |

**Definition**

Draws the specified text using the format information provided by an [SharpDX.DirectWrite.TextFormat](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textformat.htm) object.

(See also [unmanaged API documentation](http://msdn.microsoft.com/en-us/library/dd742848.aspx))

|  |
| --- |
| **Note**:  This method doesn't return an error code if it fails. |

**Method Return Value**

This method does not return a value.

**Syntax**

RenderTarget.DrawText(string text, TextFormat textFormat, RectangleF layoutRect, Brush defaultForegroundBrush)  
RenderTarget.DrawText(string text, TextFormat textFormat, RectangleF layoutRect, Brush defaultForegroundBrush, DrawTextOptions options)  
RenderTarget.DrawText(string text, TextFormat textFormat, RectangleF layoutRect, Brush defaultForegroundBrush, DrawTextOptions options,   MeasuringMode measuringMode)  
RenderTarget.DrawText(string text, int stringLength, TextFormat textFormat, RectangleF layoutRect, Brush defaultForegroundBrush, RenderTarget.DrawTextOptions options, MeasuringMode measuringMode)

**Parameters**

|  |  |
| --- | --- |
| defaultForegroundBrush | The [SharpDX.Direct2D1.Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) used to paint the text. |
| layoutRect | A [SharpDX.RectangleF](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_rectanglef.htm) which determines size and position of the area in which the text is drawn. |
| measuringMode | A [SharpDX.Direct2D1.MeasuringMode](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_measuringmode.htm) value that indicates how glyph metrics are used to measure text when it is formatted. The default value is DWRITE\_MEASURING\_MODE\_NATURAL. |
| options | A [SharpDX.Direct2D1.DrawTextOptions](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_drawtextoptions.htm) value that indicates whether the text should be snapped to pixel boundaries and whether the text should be clipped to the layout rectangle. The default value is None, which indicates that text should be snapped to pixel boundaries and it should not be clipped to the layout rectangle. |
| stringLength | An int value which represents the number of characters in string. |
| text | A string reference to an array of Unicode characters to draw. |
| textFormat | A [SharpDX.DirectWrite.TextFormat](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textformat.htm) object that describes formatting details of the text to draw, such as the font, the font size, and flow direction. |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) > [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) >  **SharpDX.Direct2D1.RenderTarget.DrawTextLayout()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawtext.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillellipse.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | | |

**Definition**

Draws the formatted text described by the specified SharpDX.DirectWrite.TextLayout object.

(See also [unmanaged API documentation](http://msdn.microsoft.com/en-us/library/dd371913.aspx))

|  |
| --- |
| **Notes**:  1.When drawing the same text repeatedly, using the **DrawTextLayout()** method is more efficient than using the [DrawText()](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawtext.htm) method because the text doesn't need to be formatted and the layout processed with each call.  2.This method doesn't return an error code if it fails. |

**Method Return Value**

This method does not return a value

**Syntax**

RenderTarget.DrawTextLayout(Vector2 origin, TextLayout textLayout, Brush defaultForegroundBrush)  
RenderTarget.DrawTextLayout(Vector2 origin, TextLayout textLayout, Brush defaultForegroundBrush, DrawTextOptions options)

**Parameters**

|  |  |
| --- | --- |
| defaultForegroundBrush | The [SharpDX.Direct2D1.Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) used to paint any text in textLayout that does not already have a brush associated with it as a drawing effect (specified by the SetDrawingEffect method). |
| options | A [SharpDX.Direct2D1.DrawTextOptions](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_drawtextoptions.htm) value that indicates whether the text should be snapped to pixel boundaries and whether the text should be clipped to the layout rectangle. The default value is None, which indicates that text should be snapped to pixel boundaries and it should not be clipped to the layout rectangle. |
| origin | A [SharpDX.Vector2](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_vector2.htm) described in device-independent pixels, at which the upper-left corner of the text described by textLayout is drawn. |
| textLayout | A [SharpDX.DirectWrite.TextLayout](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout.htm) representing the formatted text to draw. Any drawing effects that do not inherit from Resource are ignored. If there are drawing effects that inherit from ID2D1Resource that are not brushes, this method fails and the render target is put in an error state. |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) > [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) >  **SharpDX.Direct2D1.RenderTarget.FillEllipse()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_drawtextlayout.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillgeometry.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | | |

**Definition**

Paints the interior of the specified ellipse.

(See also [unmanaged API documentation](http://msdn.microsoft.com/en-us/library/dd371928.aspx))

|  |
| --- |
| **Note**:  This method doesn't return an error code if it fails. |

**Method Return Value**

This method does not return a value

**Syntax**

RenderTarget.FillEllipse(Ellipse ellipse, Brush brush)

**Parameters**

|  |  |
| --- | --- |
| brush | A [SharpDX.Direct2D1.Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) used to paint the interior of the ellipse. |
| ellipse | A [SharpDX.Direct2D1.Ellipse](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_ellipse.htm) which describes the position and radius, in device-independent pixels, of the ellipse to paint. |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX.Direct2D1](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1.htm) > [RenderTarget](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) >  **SharpDX.Direct2D1.RenderTarget.FillGeometry()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillellipse.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_rendertarget_fillrectangle.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | | |

**Definition**

Paints the interior of the specified geometry.

(See also [unamanged API documentation](http://msdn.microsoft.com/en-us/library/dd371933.aspx))

|  |
| --- |
| **Note**:  1.If the opacityBrush parameter is not null, the alpha value of each pixel of the mapped opacityBrush is used to determine the resulting opacity of each corresponding pixel of the geometry. Only the alpha value of each color in the brush is used for this processing; all other color information is ignored. The alpha value specified by the brush is multiplied by the alpha value of the geometry after the geometry has been painted by brush.  2.This method doesn't return an error code if it fails. |

**Method Return Value**

This method does not return a value.

**Syntax**

RenderTarget.FillGeometry(Geometry geometry, Brush brush)  
RenderTarget.FillGeometry(Geometry geometry, Brush brush, Brush opacityBrush)

**Parameters**

|  |  |
| --- | --- |
| brush | The [SharpDX.Direct2D1.Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) used to paint the geometry's interior. |
| geometry | The [SharpDX.Direct2D1.Geometry](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_pathgeometry.htm) to paint. |
| opacityBrush | The [SharpDX.Direct2D1.Brush](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_direct2d1_brush.htm) opacity mask to apply to the geometry, or null for no opacity mask. For more information, see the note section above |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx.htm) > [DisposeBase](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase.htm) >  **SharpDX.DisposeBase.IsDisposed** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase_dispose.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_matrix3x2.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | | |

**Definition**

Gets a value indicating whether this instance is disposed.

**Property Value**

A bool which is **true** if this instance is disposed; otherwise, **false**.

**Syntax**

<DisposeBaseObject>.IsDisposed

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) > [SharpDX](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx.htm) >  **SharpDX.Matrix3x2** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_disposebase_isdisposed.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_rectanglef.htm) |
| **Disclaimer**: The [SharpDX SDK Reference](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_sdk_reference.htm) section was compiled from the official [SharpDX Documentation](http://sharpdx.org/) and was **NOT** authored by NinjaTrader.  The contents of this section are provided as-is and only cover a fraction of what is available from the **SharpDX SDK**.  This page was intended only as a reference guide to help you get started with some of the 2D Graphics concepts used in the **NinjaTrader.Custom** assembly.  Please refer to the official **SharpDX Documentation** for additional members not covered in this reference.  For more seasoned graphic developers, the original**MSDN** [Direct2D1](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370990.aspx) and [DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038.aspx) unmanaged API documentation can also be helpful for understanding the DirectX/Direct2D run-time environment. *For NinjaScript development purposes, we document only****essential****members in the structure of this page.* | |

**Definition**

Represents a 3x2 mathematical matrix.

|  |
| --- |
| **Tip**:  For more information on Direct2D transforms, please see the [MSDN Direct2D Transforms Overview](https://msdn.microsoft.com/en-us/library/dd756655(v=vs.85).aspx) |

**Syntax**

struct Matrix3x2

**Constructors**

|  |  |
| --- | --- |
| new Matrix3x2() | Initializes a new instance of the **Matrix3x2** struct |

**Methods and Properties**

|  |  |
| --- | --- |
| Identity | Gets the identity matrix. |
| M11 | A float for the first element of the first row. |
| M12 | A float for the second element of the first row. |
| M21 | A float for the first element of the second row. |
| M22 | A float for the second element of the second row. |
| M31 | A float for the first element of the third row. |
| M32 | A float for the second element of the third row. |
| TranslationVector | A [SharpDX.Vector2](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_vector2.htm)for the translation component of this matrix. |
| Matrix3x2.Rotation(float angle) | Creates a matrix that rotates. |
| Matrix3x2.Scaling(float scale) | Creates a matrix that uniformally scales along all three axis. |
| Translation([Vector2](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_vector2.htm) value) | Creates a translation matrix using the specified offsets. |

[Ir al contenido principal](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#main)

[**Learn**](https://learn.microsoft.com/es-mx/)

* Detectar
* Documentación del producto
* Lenguajes de desarrollo
* Temas

[Iniciar sesión](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN)

[**Desarrollo de aplicaciones de Windows**](https://learn.microsoft.com/es-mx/windows/apps)

* Explorar
* Desarrollo
* Plataformas
* [Solución de problemas](https://learn.microsoft.com/es-mx/troubleshoot/windows/win32/)
* Recursos

[**Panel**](https://partner.microsoft.com/dashboard)

 Es posible que algunas partes de este tema se tradujeran mediante traducción automática o IA.

Ignorar alerta

Principio del formulario

Buscar

Final del formulario

* [Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-portal)
* [Novedades de Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/what-s-new-in-direct2d-for-windows-8-consumer-preview)
* [Acerca de Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-overview)
* [Inicio rápido de Direct2D para Windows 8](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-quickstart-with-device-context)
* Introducción con Direct2D
* Guía de programación
  + [Guía de programación](https://learn.microsoft.com/es-mx/windows/win32/direct2d/programming-guide)
  + [Introducción a la API de Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/the-direct2d-api)
  + [Direct2D y Valores altos de PPP](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-and-high-dpi)
  + [Directivas de control de errores de Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-error-handling-policies)
  + [Dispositivos y contextos de dispositivo](https://learn.microsoft.com/es-mx/windows/win32/direct2d/devices-and-device-contexts)
  + [Mejora del rendimiento de las aplicaciones de Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/improving-direct2d-performance)
  + [Introducción a las capas](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-layers-overview)
  + [Imprimir y listas de comandos](https://learn.microsoft.com/es-mx/windows/win32/direct2d/printing-and-command-lists)
  + [Aplicaciones Direct2D multiproceso](https://learn.microsoft.com/es-mx/windows/win32/direct2d/multi-threaded-direct2d-apps)
  + [Generación de perfiles de aplicaciones DirectX](https://learn.microsoft.com/es-mx/windows/win32/direct2d/profiling-directx-applications)
  + [Compresión de bloques](https://learn.microsoft.com/es-mx/windows/win32/direct2d/block-compression)
  + Efectos
  + Pinceles
  + Geometrías
  + Interoperabilidad
  + Transformaciones
    - [Transformaciones](https://learn.microsoft.com/es-mx/windows/win32/direct2d/transforms)
    - [**Información general sobre transformaciones**](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview)
    - Temas de procedimientos de transformaciones
  + Temas de procedimientos
  + [Representación de texto con Direct2D y DirectWrite](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-and-directwrite)
  + [Información general sobre las máscaras de opacidad](https://learn.microsoft.com/es-mx/windows/win32/direct2d/opacity-masks-overview)
  + [Introducción a los recursos](https://learn.microsoft.com/es-mx/windows/win32/direct2d/resources-and-resource-domains)
  + [Formatos de píxel admitidos y modos alfa](https://learn.microsoft.com/es-mx/windows/win32/direct2d/supported-pixel-formats-and-alpha-modes)
  + [Uso de Direct2D para la representación de Server-Side](https://learn.microsoft.com/es-mx/windows/win32/direct2d/server-side-rendering-overview)
  + [Información general sobre destinos de representación](https://learn.microsoft.com/es-mx/windows/win32/direct2d/render-targets-overview)
  + [Introducción a los destinos de representación de A8 compatibles](https://learn.microsoft.com/es-mx/windows/win32/direct2d/compatible-a8-rendertargets)
* Referencia de Direct2D
* Herramientas y utilidades
* [Ejemplos](https://learn.microsoft.com/es-mx/windows/win32/direct2d/d2d-samples)
* [Glosario de Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-glossary)

Descargar PDF

1. [Win32](https://learn.microsoft.com/es-mx/windows/win32/)

1. [Tecnologías de escritorio](https://learn.microsoft.com/es-mx/windows/win32/desktop-app-technologies)

1. [Gráficos y juegos](https://learn.microsoft.com/es-mx/windows/win32/graphics-and-multimedia)

1. [Gráficos y juegos de DirectX](https://learn.microsoft.com/es-mx/windows/win32/directx)

1. [Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-portal)

[**Leer en inglés**](https://learn.microsoft.com/en-us/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN)Guardar

**Información general sobre transformaciones**

* Artículo
* 12/06/2023
* 6 colaboradores

Comentarios

**En este artículo**

1. [¿Qué es una transformación de Direct2D?](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#what-is-a-direct2d-transform)
2. [Espacio de coordenadas de Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#the-direct2d-coordinate-space)
3. [Creación de matrices de transformación](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#creating-transformation-matrices)
4. [Transformaciones de destino de representación](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#rendering-target-transforms)

Mostrar 5 más

En este tema se describen los conceptos básicos de las transformaciones de Direct2D e incluye ejemplos de diversas transformaciones. Contiene las siguientes partes:

* [¿Qué es una transformación de Direct2D?](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#what-is-a-direct2d-transform)
* [Espacio de coordenadas de Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#the-direct2d-coordinate-space)
* [Creación de matrices de transformación](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#creating-transformation-matrices)
* [Transformaciones de destino de representación](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#rendering-target-transforms)
* [Transformaciones de pincel](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#brush-transforms)
* [Transformaciones de geometría](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#geometry-transforms)
* [Cómo afecta una transformación de destino de representación a los clips](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#how-a-render-target-transform-affects-clips)
* [Resumen](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#summary)
* [Temas relacionados](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-transforms-overview?redirectedfrom=MSDN#related-topics)

**¿Qué es una transformación de Direct2D?**

Una transformación especifica cómo asignar los puntos de un objeto de un espacio de coordenadas a otro o desde una posición a otra dentro del mismo espacio de coordenadas. Esta asignación se describe mediante una matriz de transformación, definida como una colección de tres filas con tres columnas de valores FLOAT, como se muestra en la tabla siguiente.

Expandir tabla

|  |  |  |
| --- | --- | --- |
| M11Default: 1.0 | M12Default: 0.0 | 0,0 |
| M21Default: 0.0 | M22Default: 1.0 | 0,0 |
| M31OffsetX: 0.0 | M32OffsetY: 0.0 | 1.0 |

En esta matriz, los miembros M11, M12, M21 y M22 definen una transformación lineal que puede escalar, girar o sesgar un objeto; Los miembros OffsetX y OffsetY definen la traducción que se aplicará después de que se haya realizado la transformación lineal. Para las transformaciones affine, los valores de la tercera columna siempre son 0,0, 0,0 y 1,0.

Dado que Direct2D solo admite transformaciones affine (lineales), su matriz de transformación se define como una matriz de 3 a 2, omitiendo la tercera columna de la matriz de transformación anterior. En la tabla siguiente se muestra el diseño de la matriz de transformación Direct2D.

Expandir tabla

|  |  |
| --- | --- |
| M11Default: 1.0 | M12Default: 0.0 |
| M21Default: 0.0 | M22Default: 1.0 |
| M31OffsetX: 0.0 | M32OffsetY: 0.0 |

En Direct2D, esta matriz de 3 a 2 se representa mediante la estructura [**D2D1\_MATRIX\_3X2**](https://learn.microsoft.com/es-es/windows/desktop/api/dcommon/ns-dcommon-d2d_matrix_3x2_f) . Para simplificar las operaciones comunes de matriz, Direct2D también proporciona una clase denominada [**Matrix3x2F**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1helper/nl-d2d1helper-matrix3x2f), que se deriva de la estructura **de D2D1\_MATRIX\_3X2** .

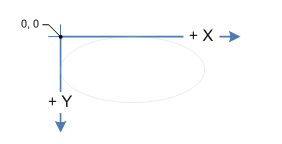
El constructor predeterminado de [**Matrix3x2F**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1helper/nl-d2d1helper-matrix3x2f) deja el objeto sin inicializar. Para recuperar una matriz de identidades, use [**Matrix3x2F::Identity**](https://learn.microsoft.com/es-es/windows/desktop/api/d2d1helper/nf-d2d1helper-identitymatrix).

Cuando se aplica una transformación de identidad a un objeto, no cambia la posición, la forma ni el tamaño del objeto. Es similar a la forma en que multiplicar un número por 1 no cambia el número. En otras palabras, la transformación de identidad deja solo las coordenadas de los puntos y no desplaza los puntos a una nueva posición. Cualquier transformación distinta de la transformación de identidad modificará la posición, la forma o el tamaño de los objetos.

Las transformaciones se refieren a las coordenadas y la comprensión del espacio de coordenadas de Direct2D es importante para comprender el uso de transformaciones.

**Espacio de coordenadas de Direct2D**

Direct2D usa un espacio de coordenadas izquierdo; es decir, los valores positivos del eje X aumentan a la derecha y los valores positivos del eje Y aumentan hacia abajo. Todo en la pantalla se coloca en relación con el origen, que es el punto en el que el eje X y el eje Y se intersecan (0, 0), como se muestra en la ilustración siguiente. Los destinos de representación de Direct2D usan este espacio de coordenadas.



Mediante la manipulación de valores en una matriz de transformación, puede girar, escalar, asimetría y mover (traducir) un objeto. Por ejemplo, si establece OffsetX en 100 y OffsetY en 200, mueve el objeto a la derecha de 100 píxeles y 200 píxeles hacia abajo.

Para mostrar el efecto del movimiento del objeto, debe aplicar la transformación de traducción para representar destinos, pinceles o geometrías. La aplicación de una transformación para representar destinos afecta a toda la pantalla, mientras que la aplicación de una transformación a un pincel o una geometría solo afecta a ese pincel o geometría específicos. Para crear una matriz de transformación, use la clase [**Matrix3x2F**](https://learn.microsoft.com/es-es/windows/desktop/api/d2d1helper/nl-d2d1helper-matrix3x2f) .

**Creación de matrices de transformación**

Para crear transformaciones de rotación, escala, asimetría y traducción, la clase [**Matrix3x2F**](https://learn.microsoft.com/es-es/windows/desktop/api/d2d1helper/nl-d2d1helper-matrix3x2f) proporciona los métodos estáticos que se muestran en la tabla siguiente. La columna Ejemplo de la tabla contiene vínculos a los temas de procedimientos que muestran cómo usar cada método de transformación.

Expandir tabla

| **Método** | **Descripción** | **Ejemplo** | **Ilustración** |
| --- | --- | --- | --- |
| [**matrix3x2f::rotate**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1helper/nf-d2d1helper-matrix3x2f-rotation) | crea una transformación de rotación que tiene el ángulo y el punto central especificados. | [cómo rotar un objeto](https://learn.microsoft.com/es-mx/windows/win32/direct2d/how-to-rotate) | ilustración de un cuadrado girado de 45 grados en el sentido de las agujas del reloj sobre el centro del cuadrado original |
| [**matrix3x2f::scale**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1helper/nf-d2d1helper-matrix3x2f-scale(d2d1_size_f_d2d1_point_2f)) | crea una transformación de escala que tiene los factores de escala y el punto central especificados. | [escalado de un objeto](https://learn.microsoft.com/es-mx/windows/win32/direct2d/how-to-scale) | ilustración de un cuadrado escalado del 130 % |
| [**matrix3x2f::skew**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1helper/nf-d2d1helper-matrix3x2f-skew) | crea una transformación de asimetría que tiene los valores de eje x e y y especificados y el punto central. | [cómo sesgar un objeto](https://learn.microsoft.com/es-mx/windows/win32/direct2d/how-to-skew) | ilustración de una asimetría cuadrada de 30 grados en sentido contrario a las agujas del reloj desde el eje Y |
| [**matrix3x2f::translation**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1helper/nf-d2d1helper-matrix3x2f-translation(d2d1_size_f)) | crea una transformación de traducción y especifica los desplazamientos en la dirección del eje X y del eje Y. | [cómo traducir un objeto](https://learn.microsoft.com/es-mx/windows/win32/direct2d/how-to-translate) | ilustración de un cuadrado movido 20 unidades a lo largo del eje X positivo y 10 unidades a lo largo del eje Y positivo |

**Transformaciones de destino de representación**

Un destino de representación es un recurso que hereda de la interfaz [**ID2D1RenderTarget**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1/nn-d2d1-id2d1rendertarget) . Crea recursos para dibujar y realiza operaciones de dibujo reales. También proporciona métodos para transformar el espacio de coordenadas. Puede llamar al método [**ID2D1RenderTarget::SetTransform**](https://learn.microsoft.com/es-mx/windows/win32/direct2d/id2d1rendertarget-settransform) para aplicar la transformación especificada al destino de representación. Todas las operaciones de dibujo posteriores se producen en el espacio transformado.

Para representar contenido, use los métodos de dibujo del destino de representación. Antes de empezar a dibujar, llame al método [**BeginDraw**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1/nf-d2d1-id2d1rendertarget-begindraw) . Para finalizar la representación del contenido, llame al método [**EndDraw**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1/nf-d2d1-id2d1rendertarget-enddraw) . Para obtener un ejemplo, vea [Cómo aplicar varias transformaciones a un objeto](https://learn.microsoft.com/es-mx/windows/win32/direct2d/how-to-apply-multiple-transforms).

**Transformaciones de pincel**

Puede ajustar la transformación en el pincel llamando a [**SetTransform**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1/nf-d2d1-id2d1brush-settransform(constd2d1_matrix_3x2_f_)). Para esta transformación, puede considerar el pincel como una gran pieza de papel y de los diferentes primitivos de representación (texto, geometría, rectángulo, etc.) como galerías de símbolos. Cuando ajustas la transformación del pincel, es como si deslizaste el papel grande debajo de la galería de símbolos, sin cambiar la posición de la galería de símbolos. Puede usar esta técnica para hacer que el texto se desvanezca de amarillo a negro en el espacio 3D.

Cuando la transformación de pincel es la transformación de identidad, los pinceles aparecen en el mismo espacio de coordenadas que el destino de representación en el que se dibujan. La transformación de pincel permite a un autor de la llamada modificar cómo se asignan las coordenadas del pincel a este espacio.

El espacio de pincel se especifica de forma diferente en Direct2D que en Windows Presentation Foundation (WPF). En Direct2D, el espacio de pincel no es relativo al objeto que se dibuja, sino que es el sistema de coordenadas actual del destino de representación, transformado por la transformación de pincel, si hay uno. Para que el pincel rellene un objeto tal como se hizo en WPF, debe traducir el origen del espacio del pincel a la esquina superior izquierda del cuadro de límite del objeto y, a continuación, escalar el espacio del pincel para que el icono base rellene el cuadro de límite del objeto.

Para obtener más información sobre las transformaciones de pincel, consulte [Información general sobre pinceles de Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-brushes-overview).

**Transformaciones de geometría**

Al escalar, mover, traducir o sesgar geometrías, puede aplicar directamente una transformación a una geometría específica, no a una transformación de destino de representación que afectaría a toda la pantalla. Una transformación de destino de representación generalmente afecta al trazo y al relleno de una geometría. Por el contrario, una transformación de geometría solo afecta al relleno de una geometría, ya que la transformación se aplica a una geometría antes de que se trazos.

**Nota**

A partir de Windows 8, la transformación del mundo no afecta al trazo si establece el tipo de trazo en [**D2D1\_STROKE\_TRANSFORM\_TYPE\_FIXED**](https://learn.microsoft.com/es-es/windows/desktop/api/D2d1_1/ne-d2d1_1-d2d1_stroke_transform_type) o [**D2D1\_STROKE\_TRANSFORM\_TYPE\_HAIRLINE**](https://learn.microsoft.com/es-es/windows/desktop/api/D2d1_1/ne-d2d1_1-d2d1_stroke_transform_type).

Puede ajustar la transformación en una geometría llamando a [**ID2D1Factory::CreateTransformedGeometry**](https://learn.microsoft.com/es-es/previous-versions/windows/desktop/legacy/dd371304(v=vs.85)) para crear un objeto [**ID2D1TransformedGeometry**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1/nn-d2d1-id2d1transformedgeometry) . Para obtener más información sobre las transformaciones de geometría, consulte [Información general sobre geometrías de Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/direct2d-geometries-overview).

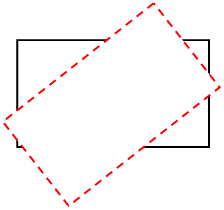
**Cómo afecta una transformación de destino de representación a los clips**

La transformación en un destino de representación afecta a cómo se calcula el rectángulo delimitador de un clip alineado con el eje. Cuando se llama a [**PushAxisAlignedClip**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1/nf-d2d1-id2d1rendertarget-pushaxisalignedclip(constd2d1_rect_f__d2d1_antialias_mode)) , el parámetro *clipRect* se transforma mediante la transformación del mundo actual que se establece en el destino de representación. Después de aplicar la transformación al *clipRect*, se calcula el rectángulo de límite alineado con el eje para *clipRect* . Para mejorar la eficacia, el contenido se recorta en este rectángulo de límite alineado con el eje y no en el *clipRect* original que se pasa. En los diagramas siguientes se muestra cómo se aplica una transformación de rotación al destino de representación, el *clipRect* resultante y un cuadro de límite alineado con el eje calculado.

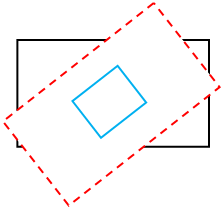
1. Supongamos que el rectángulo de la ilustración siguiente es un destino de representación que está alineado con los píxeles de pantalla.



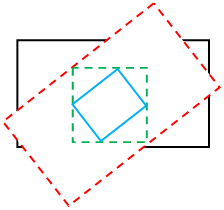
1. Aplique una transformación de rotación al destino de representación. En la ilustración siguiente, el rectángulo negro representa el destino de representación original y el rectángulo discontinuo rojo representa el destino de representación transformado.



1. Después de llamar a [**PushAxisAlignedClip**](https://learn.microsoft.com/es-es/windows/win32/api/d2d1/nf-d2d1-id2d1rendertarget-pushaxisalignedclip(constd2d1_rect_f__d2d1_antialias_mode)) , la transformación de rotación se aplica a *clipRect*. En la ilustración siguiente, el rectángulo azul representa el *clipRect* transformado.



1. Se calcula el rectángulo de límite alineado con el eje. En la ilustración siguiente, el rectángulo de guiones verde representa el rectángulo delimitador. Todo el contenido se recorta en este rectángulo de límite alineado con el eje.



**Resumen**

Direct2D facilita la transformación de objetos bidimensionales con espacios de coordenadas simplificados y clases relacionadas. Mediante el uso de varios tipos de transformaciones, puede traducir, rotar, asimetría y escalar los objetos para lograr muchos efectos visuales impresionantes.

**Temas relacionados**

[Referencia de Direct2D](https://learn.microsoft.com/es-mx/windows/win32/direct2d/reference)

**Comentarios**

**¿Le resultó útil esta página?**

SíNo

[Enviar comentarios del producto](https://www.microsoft.com/en-us/windowsinsider/feedbackhub/fb)|

[Obtén ayuda en Microsoft Q&A](https://learn.microsoft.com/answers/tags/224/windows-api-win32/)

**Recursos adicionales**

**Formación**

**Módulo**

[**Transformar digitalmente con Dynamics 365 - Training**](https://learn.microsoft.com/es-es/training/modules/digitally-transform-with-dynamics-365/?source=recommendations)

Explore cómo las aplicaciones empresariales de Dynamics 365 simplifican los sistemas CRM y ERP, ayudando a la transformación digital de las organizaciones.

**Documentación**

* [**Cómo traducir un objeto - Win32 apps**](https://learn.microsoft.com/es-es/windows/win32/direct2d/how-to-translate?source=recommendations)

Muestra cómo traducir un objeto.

* [**Información general sobre los pinceles - Win32 apps**](https://learn.microsoft.com/es-es/windows/win32/direct2d/direct2d-brushes-overview?source=recommendations)

Describe los distintos tipos de pinceles proporcionados por Direct2D.

* [**Escalado de un objeto - Win32 apps**](https://learn.microsoft.com/es-es/windows/win32/direct2d/how-to-scale?source=recommendations)

Muestra cómo escalar un objeto.

Mostrar 5 más

[**Español (México)**](https://learn.microsoft.com/es-mx/locale?target=https%3A%2F%2Flearn.microsoft.com%2Fes-mx%2Fwindows%2Fwin32%2Fdirect2d%2Fdirect2d-transforms-overview%3Fredirectedfrom%3DMSDN)

[**Sus opciones de privacidad**](https://aka.ms/yourcaliforniaprivacychoices)

Tema

* [Versiones anteriores](https://learn.microsoft.com/es-mx/previous-versions/)
* [Blog](https://techcommunity.microsoft.com/t5/microsoft-learn-blog/bg-p/MicrosoftLearnBlog)
* [Colaborar](https://learn.microsoft.com/es-mx/contribute/)
* [Privacidad](https://go.microsoft.com/fwlink/?LinkId=521839)
* [Términos de uso](https://learn.microsoft.com/es-mx/legal/termsofuse)
* [Marcas comerciales](https://www.microsoft.com/legal/intellectualproperty/Trademarks/)
* © Microsoft 2025