|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **PerformanceMetrics** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/percent.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/pips.htm) |

**Definition**

Returns a collection of custom [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm). These need to have been enabled in [Tools > Options > General](https://ninjatrader.com/es/support/helpGuides/nt8/general_section.htm) to be able to use them.

**Syntax**  
<TradeCollection>.TradesPerformance.PerformanceMetrics

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {     // Print out the number of enabled custom Performance Metrics     Print("Number of Performance Metrics: "         + SystemPerformance.AllTrades.TradesPerformance.PerformanceMetrics.Length);       // Find a the value of a specific custom Performance Metric named "MyPerformanceMetric"     for (int i = 0; i < SystemPerformance.AllTrades.TradesPerformance.PerformanceMetrics.Length; i++)     {         if (SystemPerformance.AllTrades.TradesPerformance.PerformanceMetrics[i] is               NinjaTrader.NinjaScript.PerformanceMetrics.MyPerformanceMetric)         {               Print((SystemPerformance.AllTrades.TradesPerformance.PerformanceMetrics[i] as                   NinjaTrader.NinjaScript.PerformanceMetrics.MyPerformanceMetric).Values[0]);         }     } } |

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) >  **Values** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/performanceunit.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/share_service.htm) |

**Definition**

The Values array holds an 5 values corresponding to each Cbi.PerformanceUnit. NinjaTrader will then access the Values property to display the calculated performance metric in the UI. You can also access these performance metrics for a NinjaScript strategy.

**Syntax**

public double[] Values

{ get; private set; }

**Calculating Values OnAddTrade Example**

| ns |
| --- |
| protected override void OnAddTrade(Cbi.Trade trade)  {          Values[(**int**)Cbi.PerformanceUnit.Currency] += trade.ProfitCurrency;         Values[(**int**)Cbi.PerformanceUnit.Percent]   = (1.0 + Values[(**int**)Cbi.PerformanceUnit.Percent]) \* (1.0 + trade.ProfitPercent) - 1;         Values[(**int**)Cbi.PerformanceUnit.Pips]     += trade.ProfitPips;         Values[(**int**)Cbi.PerformanceUnit.Points]   += trade.ProfitPoints;         Values[(**int**)Cbi.PerformanceUnit.Ticks]     += trade.ProfitTicks;  }    // The attribute determines the name of the performance value on the grid [Display("MyPerformanceMetric", Order = 0)] public double[] Values { get; **private** set; } |

**Calculating Values On Demand Example**

| ns | |
| --- | --- |
| // The attribute determines the name of the performance value on the grid [Display("MyPerformanceMetric", Order = 0)] public **double**[] Values {    get    {       return /\*Your custom math here\*/    }    **private** 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) > [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) >  **PerformanceUnit** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onmergeperformancemetric.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/performancemetric_values.htm) |

**Definition**

Enumeration defining each type of PerformanceUnit calculated by NinjaTrader. Used to store a value for this performance type in PerformanceMetrics.

**Syntax**

PerformanceUnit.Currency

PerformanceUnit.Percent

PerformanceUnit.Pips

PerformanceUnit.Points

PerformanceUnit.Ticks

**Examples**

| ns | |
| --- | --- |
| //Prints unrealized PnL in ticks at the close of each bar  Print(Position.GetUnrealizedProfitLoss(PerformanceUnit.Ticks, Close[0])); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) >  **PerformanceMetrics** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/orderfillresolutionvalue.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/strategy_plots.htm) |

**Definition**

Holds an array of [PerformanceMetrics](https://ninjatrader.com/es/support/helpGuides/nt8/performancemetrics.htm) objects that represent custom metrics that can be used for strategy calcuations.

Index value is based on the the array of Bars objects added via the [AddPerformanceMetric](https://ninjatrader.com/es/support/helpGuides/nt8/addperformancemetric.htm) method.

**Property Value**

An array of [PerformanceMetrics](https://ninjatrader.com/es/support/helpGuides/nt8/performancemetrics.htm) objects.

**Syntax**  
PerformanceMetrics[int index]

**Examples**

| ns | |
| --- | --- |
| // Define a new SampleCumProfit object NinjaTrader.NinjaScript.PerformanceMetrics.SampleCumProfit myProfit;   protected override void OnStateChange() {   if (State == State.Configure)   {     // Instantiate myProfit to a new instance of SampleCumProfit     myProfit = new NinjaTrader.NinjaScript.PerformanceMetrics.SampleCumProfit();           // Use AddPerformanceMetric to add myProfit to the strategy     AddPerformanceMetric(myProfit);   } }   protected override void OnBarUpdate() {   // Print a string representing the Type of the performance metric at Index 0 of the PerformanceMetrics collection   Print(PerformanceMetrics[0]); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) >  **OnMergePerformanceMetric()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/oncopyto.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/performanceunit.htm) |

**Definition**

This method is called when the Performance Metric would be aggregated and merged together (E.g. on the Strategy Analyzer's total row).

**Syntax**

protected override void OnMergePerformanceMetric(PerformanceMetricBase merge)  
{  
   
}

**Examples**

| ns | |
| --- | --- |
| protected override void OnMergePerformanceMetric(PerformanceMetricBase target) {   // You need to cast, in order to access the right type   SampleCumProfit targetMetrics = (target as SampleCumProfit);     // This is just a simple weighted average sample   if (targetMetrics != null && TradesPerformance.TradesCount + targetMetrics.TradesPerformance.TradesCount > 0)     for (int i = 0; i < Values.Length; i++)         targetMetrics.Values[i] = (targetMetrics.Values[i] \* targetMetrics.TradesPerformance.TradesCount + Values[i] \* TradesPerformance.TradesCount) / (TradesPerformance.TradesCount + targetMetrics.TradesPerformance.TradesCount); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) >  **OnMergePerformanceMetric()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/oncopyto.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/performanceunit.htm) |

**Definition**

This method is called when the Performance Metric would be aggregated and merged together (E.g. on the Strategy Analyzer's total row).

**Syntax**

protected override void OnMergePerformanceMetric(PerformanceMetricBase merge)  
{  
   
}

**Examples**

| ns | |
| --- | --- |
| protected override void OnMergePerformanceMetric(PerformanceMetricBase target) {   // You need to cast, in order to access the right type   SampleCumProfit targetMetrics = (target as SampleCumProfit);     // This is just a simple weighted average sample   if (targetMetrics != null && TradesPerformance.TradesCount + targetMetrics.TradesPerformance.TradesCount > 0)     for (int i = 0; i < Values.Length; i++)         targetMetrics.Values[i] = (targetMetrics.Values[i] \* targetMetrics.TradesPerformance.TradesCount + Values[i] \* TradesPerformance.TradesCount) / (TradesPerformance.TradesCount + targetMetrics.TradesPerformance.TradesCount); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) >  **Format()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/onaddtrade.htm) |

**Definition**

This method allows you to customize the rendering of the performance value on the Summary grid.

**Syntax**

public override string Format(object value, Cbi.PerformanceUnit unit, string propertyName)  
{  
   
}

**Examples**

| ns | |
| --- | --- |
| public override string Format(object value, Cbi.PerformanceUnit unit, string propertyName) {     double[] tmp = value as double[];     if (tmp != null && tmp.Length == 5)         switch (unit)         {               case Cbi.PerformanceUnit.Currency : return Core.Globals.FormatCurrency(tmp[0], denomination);               case Cbi.PerformanceUnit.Percent   : return tmp[1].ToString("P");               case Cbi.PerformanceUnit.Pips : return Math.Round(tmp[2]).ToString(Core.Globals.GeneralOptions.CurrentCulture);               case Cbi.PerformanceUnit.Points : return Math.Round(tmp[3]).ToString(Core.Globals.GeneralOptions.CurrentCulture);               case Cbi.PerformanceUnit.Ticks : return Math.Round(tmp[4]).ToString(Core.Globals.GeneralOptions.CurrentCulture);         }     return value.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) > [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) >  **OnAddTrade()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/format.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/oncopyto.htm) |

**Definition**

This method is called as each trade is added. You would add any custom math you wanted to do here.

|  |
| --- |
| **Note**: If your performance metric only needs to iterate through all trades at the end to perform its calculation and does not need to be calculated on each trade then using the [property approach](https://ninjatrader.com/es/support/helpGuides/nt8/performancemetric_values.htm) (On demand example) will have less of a performance impact. |

**Syntax**

**protected override void OnAddTrade(Cbi.**[**Trade**](https://ninjatrader.com/es/support/helpGuides/nt8/trade.htm)**trade)**  
**{**

**}**

**Examples**

| ns | |
| --- | --- |
| protected override void OnAddTrade(Cbi.Trade trade)  {      Values[(int)Cbi.PerformanceUnit.Currency] += trade.ProfitCurrency;      Values[(int)Cbi.PerformanceUnit.Percent]  += trade.ProfitPercent;      Values[(int)Cbi.PerformanceUnit.Pips]     += trade.ProfitPips;      Values[(int)Cbi.PerformanceUnit.Points]   += trade.ProfitPoints;      Values[(int)Cbi.PerformanceUnit.Ticks]    += trade.ProfitTicks;  } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) >  **OnCopyTo()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onaddtrade.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/onmergeperformancemetric.htm) |

**Definition**

Called as the values of a trade metric are saved.

**Syntax**

protected override void OnCopyTo(PerformanceMetricBase target)  
{  
   
}

**Examples**

| ns | |
| --- | --- |
| protected override void OnCopyTo(PerformanceMetricBase target) {   // You need to cast, in order to access the right type   SampleCumProfit targetMetrics = (target as SampleCumProfit);     if (targetMetrics != null)     Array.Copy(Values, targetMetrics.Values, Values.Length); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) >  **OnMergePerformanceMetric()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/oncopyto.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/performanceunit.htm) |

**Definition**

This method is called when the Performance Metric would be aggregated and merged together (E.g. on the Strategy Analyzer's total row).

**Syntax**

protected override void OnMergePerformanceMetric(PerformanceMetricBase merge)  
{  
   
}

**Examples**

| ns | |
| --- | --- |
| protected override void OnMergePerformanceMetric(PerformanceMetricBase target) {   // You need to cast, in order to access the right type   SampleCumProfit targetMetrics = (target as SampleCumProfit);     // This is just a simple weighted average sample   if (targetMetrics != null && TradesPerformance.TradesCount + targetMetrics.TradesPerformance.TradesCount > 0)     for (int i = 0; i < Values.Length; i++)         targetMetrics.Values[i] = (targetMetrics.Values[i] \* targetMetrics.TradesPerformance.TradesCount + Values[i] \* TradesPerformance.TradesCount) / (TradesPerformance.TradesCount + targetMetrics.TradesPerformance.TradesCount); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) >  **PerformanceUnit** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onmergeperformancemetric.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/performancemetric_values.htm) |

**Definition**

Enumeration defining each type of PerformanceUnit calculated by NinjaTrader. Used to store a value for this performance type in PerformanceMetrics.

**Syntax**

PerformanceUnit.Currency

PerformanceUnit.Percent

PerformanceUnit.Pips

PerformanceUnit.Points

PerformanceUnit.Ticks

**Examples**

| ns | |
| --- | --- |
| //Prints unrealized PnL in ticks at the close of each bar  Print(Position.GetUnrealizedProfitLoss(PerformanceUnit.Ticks, Close[0])); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) >  **Values** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/performanceunit.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/share_service.htm) |

**Definition**

The Values array holds an 5 values corresponding to each Cbi.PerformanceUnit. NinjaTrader will then access the Values property to display the calculated performance metric in the UI. You can also access these performance metrics for a NinjaScript strategy.

**Syntax**

public double[] Values

{ get; private set; }

**Calculating Values OnAddTrade Example**

| ns |
| --- |
| protected override void OnAddTrade(Cbi.Trade trade)  {          Values[(**int**)Cbi.PerformanceUnit.Currency] += trade.ProfitCurrency;         Values[(**int**)Cbi.PerformanceUnit.Percent]   = (1.0 + Values[(**int**)Cbi.PerformanceUnit.Percent]) \* (1.0 + trade.ProfitPercent) - 1;         Values[(**int**)Cbi.PerformanceUnit.Pips]     += trade.ProfitPips;         Values[(**int**)Cbi.PerformanceUnit.Points]   += trade.ProfitPoints;         Values[(**int**)Cbi.PerformanceUnit.Ticks]     += trade.ProfitTicks;  }    // The attribute determines the name of the performance value on the grid [Display("MyPerformanceMetric", Order = 0)] public double[] Values { get; **private** set; } |

**Calculating Values On Demand Example**

| ns | |
| --- | --- |
| // The attribute determines the name of the performance value on the grid [Display("MyPerformanceMetric", Order = 0)] public **double**[] Values {    get    {       return /\*Your custom math here\*/    }    **private** 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) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **AverageBarsInTrade** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/averageentryefficiency.htm) |

**Definition**

Returns the average number of bars per trade.

**Property Value**

A double value that represents the average number of bars per trade.

**Syntax**  
<TradeCollection>.TradesPerformance.AverageBarsInTrade

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the average number of bars per trade of all trades     Print("Average # bars per trade is: " + SystemPerformance.AllTrades.TradesPerformance.AverageBarsInTrade); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **AverageEntryEfficiency** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/averagebarsintrade.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/averageexitefficiency.htm) |

**Definition**

Returns the average entry efficiency.

**Property Value**

A double value that represents the average entry efficiency.

**Syntax**  
<TradeCollection>.TradesPerformance.AverageEntryEfficiency

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the average entry efficiency     Print("Average entry efficiency is: " + SystemPerformance.AllTrades.TradesPerformance.AverageEntryEfficiency); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **AverageTimeInMarket** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/averageexitefficiency.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/averagetotalefficiency.htm) |

**Definition**

Returns the average duration of a trade weighted by quantity.

**Property Value**

A TimeSpan value that represents the quantity-weighted average duration of a trade.

**Syntax**  
<TradeCollection>.TradesPerformance.AverageTimeInMarket

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the quantity-weighted average duration of all trades     Print("Average time in market: " + SystemPerformance.AllTrades.TradesPerformance.AverageTimeInMarket); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **AverageTimeInMarket** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/averageexitefficiency.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/averagetotalefficiency.htm) |

**Definition**

Returns the average duration of a trade weighted by quantity.

**Property Value**

A TimeSpan value that represents the quantity-weighted average duration of a trade.

**Syntax**  
<TradeCollection>.TradesPerformance.AverageTimeInMarket

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the quantity-weighted average duration of all trades     Print("Average time in market: " + SystemPerformance.AllTrades.TradesPerformance.AverageTimeInMarket); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **AverageTotalEfficiency** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/averagetimeinmarket.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/currency.htm) |

**Definition**

Returns the average total efficiency.

**Property Value**

A double value that represents the average total efficiency.

**Syntax**  
<TradeCollection>.TradesPerformance.AverageTotalEfficiency

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the average total efficiency     Print("Average total efficiency is: " + SystemPerformance.AllTrades.TradesPerformance.AverageTotalEfficiency); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **TotalCommission** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ticks.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/totalquantity.htm) |

**Definition**

Returns the total commission.

**Property Value**

A double value that represents the total commission.

**Syntax**  
<TradeCollection>.TradesPerformance.TotalCommission

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {     // Print out the total commission of all trades     Print("Total commission is: " + SystemPerformance.AllTrades.TradesPerformance.TotalCommission); } |

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **Currency** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/averagetotalefficiency.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/grossloss.htm) |

**Definition**

Returns a [TradesPerformanceValues](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformancevalues.htm) object in currency.

**Property Value**

A TradesPerformanceValues object that is represented in currency.

**Syntax**  
<TradeCollection>.TradesPerformance.Currency

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the avg. profit of all trades in currency     Print("Average profit: " + SystemPerformance.AllTrades.TradesPerformance.Currency.AverageProfit); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **GrossLoss** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/currency.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/grossprofit.htm) |

**Definition**

Returns the gross loss.

**Property Value**

A double value that represents the gross loss.

**Syntax**  
<TradeCollection>.TradesPerformance.GrossLoss

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the gross loss of all trades     Print("Gross loss is: " + SystemPerformance.AllTrades.TradesPerformance.GrossLoss); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **GrossProfit** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/grossloss.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/longestflatperiod.htm) |

**Definition**

Returns the gross profit.

**Property Value**

A double value that represents the gross profit.

**Syntax**  
<TradeCollection>.TradesPerformance.GrossProfit

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the gross profit of all trades     Print("Gross profit is: " + SystemPerformance.AllTrades.TradesPerformance.GrossProfit); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **LongestFlatPeriod** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/grossprofit.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/maxconsecutiveloser.htm) |

**Definition**

Returns the longest duration of being flat.

**Property Value**

A TimeSpan value that represents the longest duration of being flat.

**Syntax**  
<TradeCollection>.TradesPerformance.LongestFlatPeriod

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the longest duration of being flat     Print("Longest flat period: " + SystemPerformance.AllTrades.TradesPerformance.LongestFlatPeriod); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **MaxConsecutiveLoser** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/longestflatperiod.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/maxconsecutivewinner.htm) |

**Definition**

Returns the maximum number of consecutive losers seen.

**Property Value**

An int value that represents the maximum number of consecutive losers seen.

**Syntax**  
<TradeCollection>.TradesPerformance.MaxConsecutiveLoser

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the max consecutive losers of all trades     Print("Max # of consecutive losers is: " + SystemPerformance.AllTrades.TradesPerformance.MaxConsecutiveLoser); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **MaxConsecutiveWinner** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/maxconsecutiveloser.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/maxtimetorecover.htm) |

**Definition**

Returns the maximum number of consecutive winners seen.

**Property Value**

An int value that represents the maximum number of consecutive winners seen.

**Syntax**  
<TradeCollection>.TradesPerformance.MaxConsecutiveWinner

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the max consecutive winners of all trades     Print("Max # of consecutive winners is: " + SystemPerformance.AllTrades.TradesPerformance.MaxConsecutiveWinner); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **MaxTimeToRecover** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/maxconsecutivewinner.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/monthlystddev.htm) |

**Definition**

Returns the maximum time to recover from a draw down.

**Property Value**

A TimeSpan value that represents the maximum time to recover from a draw down.

**Syntax**  
<TradeCollection>.TradesPerformance.MaxTimeToRecover

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the maximum time to recover from a draw down     Print("Max time to recover is: " + SystemPerformance.AllTrades.TradesPerformance.MaxTimeToRecover); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **MonthlyStdDev** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/maxtimetorecover.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/monthlyulcer.htm) |

**Definition**

Returns the monthly standard deviation.

**Property Value**

A double value that represents the monthly standard deviation.

**Syntax**  
<TradeCollection>.TradesPerformance.MonthlyStdDev

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the monthly standard deviation     Print("Monthly standard deviation is: " + SystemPerformance.AllTrades.TradesPerformance.MonthlyStdDev); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **MonthlyUlcer** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/monthlystddev.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/netprofit.htm) |

**Definition**

Returns the monthly Ulcer index.

**Property Value**

A double value that represents the monthly Ulcer index.

**Syntax**  
<TradeCollection>.TradesPerformance.MonthlyUlcer

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the monthly Ulcer index     Print("Monthly Ulcer index is: " + SystemPerformance.AllTrades.TradesPerformance.MonthlyUlcer); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **NetProfit** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/monthlyulcer.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/percent.htm) |

**Definition**

Returns the net profit.

**Property Value**

A double value that represents the net profit.

**Syntax**  
<TradeCollection>.TradesPerformance.NetProfit

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the net profit of all trades     Print("Net profit is: " + SystemPerformance.AllTrades.TradesPerformance.NetProfit); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **PerformanceMetrics** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/percent.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/pips.htm) |

**Definition**

Returns a collection of custom [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm). These need to have been enabled in [Tools > Options > General](https://ninjatrader.com/es/support/helpGuides/nt8/general_section.htm) to be able to use them.

**Syntax**  
<TradeCollection>.TradesPerformance.PerformanceMetrics

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {     // Print out the number of enabled custom Performance Metrics     Print("Number of Performance Metrics: "         + SystemPerformance.AllTrades.TradesPerformance.PerformanceMetrics.Length);       // Find a the value of a specific custom Performance Metric named "MyPerformanceMetric"     for (int i = 0; i < SystemPerformance.AllTrades.TradesPerformance.PerformanceMetrics.Length; i++)     {         if (SystemPerformance.AllTrades.TradesPerformance.PerformanceMetrics[i] is               NinjaTrader.NinjaScript.PerformanceMetrics.MyPerformanceMetric)         {               Print((SystemPerformance.AllTrades.TradesPerformance.PerformanceMetrics[i] as                   NinjaTrader.NinjaScript.PerformanceMetrics.MyPerformanceMetric).Values[0]);         }     } } |

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **Pips** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/performancemetrics.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/points.htm) |

**Definition**

Returns a [TradesPerformanceValues](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformancevalues.htm) object in pips.

**Property Value**

A TradesPerformanceValues object that is represented in pips.

**Syntax**  
<TradeCollection>.TradesPerformance.Pips

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {     // Print out the avg. profit of all trades in pips     Print("Average profit: " + SystemPerformance.AllTrades.TradesPerformance.Pips.AverageProfit); } |

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **Points** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/pips.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/profitfactor.htm) |

**Definition**

Returns a [TradesPerformanceValues](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformancevalues.htm) object in points.

**Property Value**

A TradesPerformanceValues object that is represented in points.

**Syntax**  
<TradeCollection>.TradesPerformance.Points

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {     // Print out the avg. profit of all trades in points     Print("Average profit: " + SystemPerformance.AllTrades.TradesPerformance.Points.AverageProfit); } |

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) > [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) > [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) >  **ProfitFactor** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/points.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/rsquared.htm) |

**Definition**

Returns the profit factor.

**Property Value**

A double value that represents the profit factor.

**Syntax**  
<TradeCollection>.TradesPerformance.ProfitFactor

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {     // Print out the profit factor of all trades     Print("Profit factor is: " + SystemPerformance.AllTrades.TradesPerformance.ProfitFactor); } |

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

The methods and properties covered in this section are unique to custom strategy development.

**In this section**

|  |  |
| --- | --- |
| [Account](https://ninjatrader.com/es/support/helpGuides/nt8/strategy_account.htm) | Represents the real-world or simulation **Account** configured for the strategy. |
| [AddChartIndicator()](https://ninjatrader.com/es/support/helpGuides/nt8/addchartindicator.htm) | Adds an indicator to the strategy only for the purpose of displaying it on a chart. |
| [AddPerformanceMetric()](https://ninjatrader.com/es/support/helpGuides/nt8/addperformancemetric.htm) | Adds an instance of custom [Performance Metric](https://ninjatrader.com/es/support/helpGuides/nt8/performancemetrics.htm) to a strategy used in strategy calculations. |
| [ATM Strategy Methods](https://ninjatrader.com/es/support/helpGuides/nt8/atm_strategy_methods.htm) | Adds ATM strategies to manage your position |
| [BarsRequiredToTrade](https://ninjatrader.com/es/support/helpGuides/nt8/barsrequiredtotrade.htm) | The number of historical bars required before the strategy starts processing order methods called in the [OnBarUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onbarupdate.htm) method. |
| [BarsSinceEntryExecution()](https://ninjatrader.com/es/support/helpGuides/nt8/barssinceentryexecution.htm) | Returns the number of bars that have elapsed since the last specified entry. |
| [BarsSinceExitExecution()](https://ninjatrader.com/es/support/helpGuides/nt8/barssinceexitexecution.htm) | Returns the number of bars that have elapsed since the last specified exit. |
| [ChartIndicators](https://ninjatrader.com/es/support/helpGuides/nt8/chartindicators.htm) | Contains a collection of Indicators which have been added to the strategy instance using [AddChartIndicator()](https://ninjatrader.com/es/support/helpGuides/nt8/addchartindicator.htm). |
| [CloseStrategy()](https://ninjatrader.com/es/support/helpGuides/nt8/closestrategy.htm) | Cancels all working orders, closes any existing positions, and finally disables the strategy. |
| [ConnectionLossHandling](https://ninjatrader.com/es/support/helpGuides/nt8/connectionlosshandling.htm) | Sets the manner in which your strategy will behave when a connection loss is detected. |
| [DaysToLoad](https://ninjatrader.com/es/support/helpGuides/nt8/daystoload.htm) | Determines the number of trading days which will be configured when loading the strategy from the **Strategies Grid**. |
| [DefaultQuantity](https://ninjatrader.com/es/support/helpGuides/nt8/defaultquantity.htm) | An order size variable that can be set either programmatically or overriden via the Strategy that determines the quantity of an entry order. |
| [DisconnectDelaySeconds](https://ninjatrader.com/es/support/helpGuides/nt8/disconnectdelayseconds.htm) | Determines the amount of time a disconnect would have to last before [connection loss handling](https://ninjatrader.com/es/support/helpGuides/nt8/connectionlosshandling.htm) takes action. |
| [EntriesPerDirection](https://ninjatrader.com/es/support/helpGuides/nt8/entriesperdirection.htm) | Determines the maximum number of entries allowed per direction while a position is active based on the [EntryHandling](https://ninjatrader.com/es/support/helpGuides/nt8/entryhandling.htm) property. |
| [EntryHandling](https://ninjatrader.com/es/support/helpGuides/nt8/entryhandling.htm) | Sets the manner in how entry orders will handle. |
| [Execution](https://ninjatrader.com/es/support/helpGuides/nt8/execution.htm) | Represents a read only interface that exposes information regarding an execution (filled order) resulting from an order and is passed as a parameter in the [OnExecutionUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onexecutionupdate.htm) method. |
| [ExitOnSessionCloseSeconds](https://ninjatrader.com/es/support/helpGuides/nt8/exitonsessioncloseseconds.htm) | The number of seconds before the actual session end time that the "[IsExitOnSessionCloseStrategy](https://ninjatrader.com/es/support/helpGuides/nt8/isexitonsessionclosestrategy.htm)" function will trigger. |
| [IncludeCommission](https://ninjatrader.com/es/support/helpGuides/nt8/includecommission.htm) | Determines if the strategy performance results will include commission on a historical backtest. |
| [IncludeTradeHistoryInBacktest](https://ninjatrader.com/es/support/helpGuides/nt8/includetradehistoryinbacktest.htm) | Determines if the strategy will save orders, trades, and execution history. |
| [IsAdoptAccountPositionAware](https://ninjatrader.com/es/support/helpGuides/nt8/isadoptaccountpositionaware.htm) | Determines if the strategy is programmed in a manner capable of handling  real-world account positions. |
| [IsExitOnSessionCloseStrategy](https://ninjatrader.com/es/support/helpGuides/nt8/isexitonsessionclosestrategy.htm) | Determines if the strategy will cancel all strategy generated orders and close all open strategy positions at the close of the session. |
| [IsFillLimitOnTouch](https://ninjatrader.com/es/support/helpGuides/nt8/isfilllimitontouch.htm) | Determines if the strategy will use a more liberal fill algorithm for back-testing purposes only. |
| [IsInstantiatedOnEachOptimizationIteration](https://ninjatrader.com/es/support/helpGuides/nt8/isinstantiatedoneachoptimizationiteration.htm) | Determines if the strategy should be re-instantiated (re-created) after each optimization run when using the [Strategy Analyzer Optimizer](https://ninjatrader.com/es/support/helpGuides/nt8/optimize_a_strategy.htm). |
| [IsTradingHoursBreakLineVisible](https://ninjatrader.com/es/support/helpGuides/nt8/istradinghoursbreaklinevisible.htm) | Plots trading hours break lines on the indicator panel. |
| [IsWaitUntilFlat](https://ninjatrader.com/es/support/helpGuides/nt8/iswaituntilflat.htm) | Indicates the strategy is currently waiting until a flat position is detected before submitting live orders. |
| [NumberRestartAttempts](https://ninjatrader.com/es/support/helpGuides/nt8/numberrestartattempts.htm) | Determines the maximum number of restart attempts allowed within the last x minutes defined in [RestartsWithinMinutes](https://ninjatrader.com/es/support/helpGuides/nt8/restartswithinminutes.htm) when the strategy experiences a connection loss. |
| [OnAccountItemUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onaccountitemupdate.htm) | An event driven method used for strategies which is called for each AccountItem update for the account on which the strategy is running. |
| [OnExecutionUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onexecutionupdate.htm) | An event driven method which is called on an incoming execution of an order managed by a strategy. |
| [OnOrderTrace()](https://ninjatrader.com/es/support/helpGuides/nt8/onordertrace.htm) | An event driven method used for strategies which will allow you to customize the output of [TraceOrders](https://ninjatrader.com/es/support/helpGuides/nt8/traceorders.htm). |
| [OnOrderUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onorderupdate.htm) | An event driven method which is called each time an order managed by a strategy changes state. |
| [OnPositionUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onpositionupdate.htm) | An event driven method which is called each time the position of a strategy changes state. |
| [OptimizationPeriod](https://ninjatrader.com/es/support/helpGuides/nt8/order.htm) | Reserved for [Walk-Forward Optimization](https://ninjatrader.com/es/support/helpGuides/nt8/walk_forward_optimize_a_strate.htm), this property determines the number of days used for the "in sample" backtest period for a given strategy.  See also [TestPeriod](https://ninjatrader.com/es/support/helpGuides/nt8/testperiod.htm). |
| [Order](https://ninjatrader.com/es/support/helpGuides/nt8/order.htm) | Represents a read only interface that exposes information regarding an order. |
| [Order Methods](https://ninjatrader.com/es/support/helpGuides/nt8/order_methods.htm) | NinjaScript provides several approaches you can use for order placement within your NinjaScript strategy. |
| [OrderFillResolution](https://ninjatrader.com/es/support/helpGuides/nt8/orderfillresolution.htm) | Determines how strategy orders are filled during historical states. |
| [OrderFillResolutionType](https://ninjatrader.com/es/support/helpGuides/nt8/orderfillresolutiontype.htm) | Determines the bars type which will be used for historical fill processing. |
| [OrderFillResolutionValue](https://ninjatrader.com/es/support/helpGuides/nt8/orderfillresolutionvalue.htm) | Determines the bars period interval value which will be used for historical fill processing. |
| [PerformanceMetrics](https://ninjatrader.com/es/support/helpGuides/nt8/strategy_performancemetrics.htm) | Holds an array of [PerformanceMetrics](https://ninjatrader.com/es/support/helpGuides/nt8/performancemetrics.htm) objects that represent custom metrics that can be used for strategy calcuations. |
| [Plots](https://ninjatrader.com/es/support/helpGuides/nt8/strategy_plots.htm) | A collection holding all of the Plot objects that define their visualization characteristics. |
| [Position](https://ninjatrader.com/es/support/helpGuides/nt8/position.htm) | Represents position related information that pertains to an instance of a strategy. |
| [PositionAccount](https://ninjatrader.com/es/support/helpGuides/nt8/positionaccount.htm) | Represents position related information that pertains to real-world account (live or simulation). |
| [Positions](https://ninjatrader.com/es/support/helpGuides/nt8/positions.htm) | Holds an array of [Position](https://ninjatrader.com/es/support/helpGuides/nt8/position.htm) objects that represent positions managed by the strategy. |
| [PositionsAccount](https://ninjatrader.com/es/support/helpGuides/nt8/positionsaccount.htm) | Holds an array of [PositionAccount](https://ninjatrader.com/es/support/helpGuides/nt8/positionaccount.htm) objects that represent positions managed by the strategy's account. |
| [RealtimeErrorHandling](https://ninjatrader.com/es/support/helpGuides/nt8/realtimeerrorhandling.htm) | Defines the behavior of a strategy when a strategy generated order is returned from the broker's server in a "Rejected" state. |
| [RestartsWithinMinutes](https://ninjatrader.com/es/support/helpGuides/nt8/restartswithinminutes.htm) | Determines within how many minutes the strategy will attempt to restart. |
| [SetOrderQuantity](https://ninjatrader.com/es/support/helpGuides/nt8/setorderquantity.htm) | Determines how order sizes are calculated for a given strategy. |
| [Slippage](https://ninjatrader.com/es/support/helpGuides/nt8/slippage.htm) | Sets the amount of slippage in ticks per execution used in performance calculations during backtests. |
| [StartBehavior](https://ninjatrader.com/es/support/helpGuides/nt8/startbehavior.htm) | Sets the start behavior of the strategy. See [Syncing Account Positions](https://ninjatrader.com/es/support/helpGuides/nt8/syncing_account_positions.htm) for more information. |
| [StopTargetHandling](https://ninjatrader.com/es/support/helpGuides/nt8/stoptargethandling.htm) | Determines how stop and target orders are submitted during an entry order execution. |
| [StrategyBaseConverter](https://ninjatrader.com/es/support/helpGuides/nt8/strategybaseconverter.htm) | A custom TypeConverter class handling the designed behavior of an strategy's property descriptor collection. |
| [SystemPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/systemperformance.htm) | The SystemPerformance object holds all trades and trade performance data generated by a strategy. |
| [TestPeriod](https://ninjatrader.com/es/support/helpGuides/nt8/testperiod.htm) | Reserved for [Walk-Forward Optimization](https://ninjatrader.com/es/support/helpGuides/nt8/walk_forward_optimize_a_strate.htm), this property determines the number of days used for the "out of sample" backtest period for a given strategy. |
| [TimeInForce](https://ninjatrader.com/es/support/helpGuides/nt8/timeinforce.htm) | Sets the time in force property for all orders generated by a strategy. |
| [TraceOrders](https://ninjatrader.com/es/support/helpGuides/nt8/traceorders.htm) | Determines if OnOrderTrace() would be called for a given strategy. |
| [Trade](https://ninjatrader.com/es/support/helpGuides/nt8/trade.htm) | A Trade is a completed buy/sell or sell/buy transaction. It consists of an entry and exit execution. |
| [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) | A collection of [Trade](https://ninjatrader.com/es/support/helpGuides/nt8/trade.htm) objects. |
| [TradesPerformanceValues](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformancevalues.htm) | Performance values of a [collection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) of [Trade](https://ninjatrader.com/es/support/helpGuides/nt8/trade.htm) objects. |
| [WaitForOcoClosingBracket](https://ninjatrader.com/es/support/helpGuides/nt8/waitforococlosingbracket.htm) | Determines if the strategy will submit both legs of an OCO bracket before submitting the pair to the broker. |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) >  **AddPerformanceMetric()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/addchartindicator.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/atm_strategy_methods.htm) |

**Definition**

Adds an instance of custom [Performance Metric](https://ninjatrader.com/es/support/helpGuides/nt8/performancemetrics.htm) to a strategy used in strategy calculations.

**Method Return Value**

This method does not return a value.

**Syntax**  
AddPerformanceMetric(PerformanceMetricBase performanceMetric)

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

**Parameters**

|  |  |
| --- | --- |
| performanceMetric | The performance metric object to be added |

**Examples**

| ns |
| --- |
| protected override void OnStateChange() {     if (State == State.Configure)     {         AddPerformanceMetric(new NinjaTrader.NinjaScript.PerformanceMetrics.SampleCumProfit());     } } |

|  |  |
| --- | --- |
| **Navigation:**  »No topics above this level«  **8.0.0.9 (Beta) Release Notes** | [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout.htm) |

**Release Date**

February 16, 2016

This is a general maintenance release which includes various bug fixes due to feedback received during beta. Please note that as this is a maintenance release, there are still outstanding bugs which were reported in previous beta versions and this release does not guarantee all previously reported bugs have been resolved.  If you cannot find the status of a bug you previously reported using the table below, please feel free to contact our support team with the tracking ID which was provided at the time the bug was reported (e.g., NTEIGHT-1234)

|  |
| --- |
| **Attention Trading Technologies Users:**Due to limited use and low user feedback during the beta period, we have removed the **Trading Technologies** adapter from NinjaTrader 8.Users can still connect to their brokerage by using NinjaTrader 7. |

|  |
| --- |
| **Attention FXCM Users:**As a result of Issue #9270, related to FXCM CFD's and commissions, you may need to manually delete your database file if you were using an **FXCM** **CFD** account with a **Commission Template** configured, otherwise errors may be generated.  Please contact [platformsupport@ninjatrader.com](mailto:platformsupport@ninjatrader.com) should you receive errors on start-up after updating. |

**Code Breaking Changes**

|  |
| --- |
| **Tip**:  We have expanded information in our NinjaScript**Educational Resources** section regrading best practices for new [muti-threading considerations](https://ninjatrader.com/es/support/helpGuides/nt8/multi-threading.htm) in NinjaTrader 8 |

•[**SetTrailStop()**](https://ninjatrader.com/es/support/helpGuides/nt8/settrailstop.htm)**,**[**SetStopLoss()**](https://ninjatrader.com/es/support/helpGuides/nt8/setstoploss.htm)**,**[**SetProfitTarget()**](https://ninjatrader.com/es/support/helpGuides/nt8/setprofittarget.htm)- removed redundant doubletype "***currency***" overloaded.  Set **CalculationMode** enum instead e.g., *SetTrailStop(CalculationMode.Price, Low[0])*

•[**DrawingTools.ChartAnchor**](https://ninjatrader.com/es/support/helpGuides/nt8/chartanchor.htm)- removed **ChartAnchor** constructor with int type "***barsAgo"*** parameter as it was problematic.  Use **DateTime** ***time*** values to build your drawing tools chart anchor objects instead.

•**Bars.SessionIterator** object was removed due to complications with some bar types.  You can use your own custom [SessionIterator](https://ninjatrader.com/es/support/helpGuides/nt8/sessioniterator.htm) built from a **Bars** object instead.

a.For **Indicators,**please see the system Custom\Indicators*@CurrentDayOHL.cs, @Pivots.cs, PriorDayOHLC.cs*, or the examples in the [SessionIterator](https://ninjatrader.com/es/support/helpGuides/nt8/sessioniterator.htm) Help guide section for examples of correct indicator usage.

b.For custom **Bar Types**, please see the [BarsType.SessionIterator](https://ninjatrader.com/es/support/helpGuides/nt8/barstype_sessioniterator.htm) Help Guide page, or any of the system Custom\BarTypes installed by default.

•There were several changes to **PerformanceMetrics**.  These changes will **NOT** affect any strategy/system performance calculations, but if you develop own custom [PerformanceMetric](https://ninjatrader.com/es/support/helpGuides/nt8/performancemetrics.htm) please be aware of the changes listed below.  You may also review the **PerformanceMetric***@SampleCumProfit.cs*for an example of correct usage.

a.**PerformanceMetric.ValueArrayLength** was removed as it is no longer need

b.Added new [OnCopyTo()](https://ninjatrader.com/es/support/helpGuides/nt8/oncopyto.htm)method

c.[OnMergePerformanceMetric()](https://ninjatrader.com/es/support/helpGuides/nt8/onmergeperformancemetric.htm) has a reversed the target/source logic to be in sync with new **OnCopyTo()** method

**Notes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Status** | **Issue #** | **Category** | **Comments** |
| Changed | 5734 | UI | Tooltips in property grids: Disabled full text tip as entire text is visible |
| Added | 9319 | UI | Add undocumented string PersistenceId GUID to NTTabControl/NTTabPage |
| Fixed | 9373 | Core, Playback | The simulator option 'Enforce partial fills' no longer produces random partial fills |
| Changed | 8110 | Property Grids, UI | Property Grid Background Color and Styling |
| Fixed | 8773 | TD AMERITRADE | TD Ameritrade sometimes lost connection immediately after connecting |
| Fixed | 9044 | Indicator, NinjaScript | BarTimer was not always updating every second |
| Fixed | 9045 | Interactive Brokers | IB Gateway was not always properly restoring connection after disconnect |
| Fixed | 9082 | SuperDOM | SuperDOM Volume column did not always include volume from the previous session. |
| Fixed | 9178 | TD AMERITRADE | Requesting Index instruments could result in errors |
| Fixed | 9207 | Indicator | Indicators should now behave identical independent of IsRemoveLastBarSupported |
| Fixed | 9208 | Adapter | Fixed issues with Kinetick EOD ConnectionLost Orange and Red Connection Colors |
| Fixed | 9211 | ShareAdapter | StockTwits share adapter did not successfully posting images |
| Fixed | 9234 | Chart, Strategy | Strategy would be disabled when second strategy on multiseries chart was enabled |
| Fixed | 9235 | Tick Replay | Tick Replay Was Not Providing Bid/Ask Properly Prior to Rollover with Merge Back Adjusted |
| Fixed | 9236 | Market Analyzer | Market Analyzer "Grid Foreground" was not saving to workspace |
| Added | 9242 | Trading Hours | Added Trading Date holiday check on applying trading hour definition |
| Fixed | 9243 | Database | Custom Futures instruments migrated to NT8 switched rollover date and contract month values |
| Fixed | 9244 | Indicator | CandlestickPattern text did not load correctly |
| Fixed | 9247 | Chart, Workspaces | Chart object Z-Order was not saving with workspaces |
| Fixed | 9250 | Chart, DrawingTool | Draw Markers Were Not Rendering in Second Panel |
| Fixed | 9251 | NinjaScript | Bars.PercentComplete was not matching NT7 expected output in Market Replay |
| Fixed | 9252 | NinjaScript | Indicator wrappers could inadvertently create multiple instances of the 'same' hosted indicator |
| Fixed | 9253 | Bars, Chart | Daily Bars were not applying days to load until reload |
| Fixed | 9254 | DrawingTool, NinjaScript | Draw.TextFixed "Template" overload was not loading saved template |
| Fixed | 9255 | Quantity Selector | Quantity Selector was incorrectly rounding some values |
| Fixed | 9257 | Control Center | Email Support... with attached image file was not working |
| Fixed | 9258 | Indicator | SampleCustomPlot rectangle was shrinking incorrectly |
| Fixed | 9259 | Position Display | Position Display Points Mode was not using FormatPrice() |
| Fixed | 9260 | Backup & Restore, NinjaScript | Could export file in physical sub-folder |
| Fixed | 9261 | NinjaScript Editor | Could not create folders if a file of the same name exists |
| Fixed | 9262 | NinjaScript | Drawing objects were not appearing at correct time when drawn by indicator |
| Fixed | 9265 | NinjaScript | SetTrailStop with CalculationMode.Currency was causing ignored orders |
| Fixed | 9267 | Chart, Workspaces | Restoring workspace was not applying Show Tabs chart property |
| Fixed | 9268 | Playback, UI | Replay Controller did not  remember the last start date used |
| Fixed | 9269 | Market Analyzer | Total row was not adding total |
| Fixed | 9270 | Adapter, Commissions | FXCM CFD orders/executions/positions not displaying when using commissions |
| Fixed | 9271 | Chart, Strategy Analyzer | AddChartIndicator() did  not plot indicator in strategy analyzer chart |
| Fixed | 9274 | DrawingTool | Incorrect Text location of Fibonacci Retracement |
| Fixed | 9276 | DrawingTool, Playback | Draw.TextFixed() disappeared from chart after moving Playback slider |
| Fixed | 9277 | NinjaScript | DrawObjects.Count not updating correctly during State.Historical |
| Fixed | 9278 | Strategy Analyzer | Walk Forward Optimization Results Exported to Excel Were Poorly Formatted |
| Fixed | 9281 | Core | Simulator Cash Value incorrectly subtracted commissions |
| Fixed | 9282 | NinjaScript | Gui.CategoryOrder was not applying in Strategy Analyzer |
| Fixed | 9283 | NinjaScript | IsFirstBarOfSession not triggering in PNF charts |
| Fixed | 9284 | Chart | Y-axis unexpectedly compressed after moving the panel |
| Fixed | 9285 | Data Grids | Unstacked orders in Basic Entry grid remained displayed in the grid after being cancelled |
| Fixed | 9286 | Strategy Analyzer | Custom performance metrics not initially populating after optimization |
| Fixed | 9287 | Chart, Templates | Chart template were not restoring panel sizes on first load after chart created |
| Fixed | 9288 | NinjaScript, UI | Draw.Region() was not always rendering |
| Fixed | 9289 | Strategy Analyzer | Optimization graph results were offset by 1 when strategy utilizes a property of type string |
| Fixed | 9290 | Attach Order To Indicator | Space at beginning of indicator name prevented order's 'Attach to indicator' function from working |
| Fixed | 9291 | SuperDOM | SuperDOM was not using Indicator name defined in code |
| Fixed | 9292 | Alerts | Alerts within tabs were only triggering if tab is focused |
| Fixed | 9293 | Alerts, Property Grids | Alerts Config Window has sometimes had property line through entire grid |
| Fixed | 9294 | Alerts | Resolved threading conflict with alerts in different tabs when connecting to data provider |
| Fixed | 9296 | BarsType | PnF charts with break at EOD unchecked showed bars with all the same time |
| Fixed | 9298 | DrawingTool, Templates | Actual text in text drawing object saving/restoring with template |
| Fixed | 9299 | Alerts | Horizontal Line Alerts was not triggering when using Rearm: OnBarClose |
| Fixed | 9300 | DrawingTool | Draw.TextFixed() objects were drawn out of order |
| Fixed | 9301 | Bars | Bars.SessionIterator no longer accessible in NinjaScript |
| Fixed | 9303 | Playback | Playback controller freezing after twice connect/enable strategy/disconnect |
| Fixed | 9306 | Skins | Left hand bar in Strategy Wizard on Slate Grey Skins not colored correctly |
| Fixed | 9307 | Backup & Restore | On exit, NT8 prompts to save Backup before prompting to save Workspace |
| Fixed | 9309 | Data Grids | German decimal missing in exported excel grid |
| Fixed | 9310 | Strategy | Strategy was not added to chart when hits exception on historical data |
| Fixed | 9311 | Backup & Restore | File Dialog should now maintain the directory where last navigate |
| Fixed | 9314 | Database, Playback | Wrong 'from' and 'to' date was triggered on locally updating rollovers while connected to playback |
| Fixed | 9316 | NinjaScript | Opening data box throws exception from backtest when strategy using AddPlot() |
| Fixed | 9320 | DrawingTool | Newly generated drawing tool would freeze chart |
| Fixed | 9324 | DrawingTool | PriceLevels not were working In Compiled Assemblies |
| Fixed | 9326 | DrawingTool | DrawingTool.IsSeperateZOrder was not working as intended |
| Fixed | 9328 | NinjaScript | Draw objects from AddChartIndicator() were not drawn on SA chart |
| Fixed | 9329 | NinjaScript | Space before namespace in script causes crash when exporting to compiled assembly |
| Fixed | 9331 | Interactive Brokers | IB Linked Account were not connecting due to race condition |
| Fixed | 9333 | NinjaScript | Exporting compiled assembly that contains a class in custom namespace failed to import |
| Fixed | 9334 | NinjaScript | Exporting compiled assembly containing DateTime property failed to import |
| Fixed | 9337 | Strategy Analyzer | Strategy Analyzer crashed on closing second tab |
| Fixed | 9343 | Indicator | NT8 Pivots indicator did not match NT7's values |
| Fixed | 9344 | Chart, DrawingTool | Indicator draw objects disappeared when attempt to move series |
| Fixed | 9345 | NinjaScript | Code Wizard for strategy setting incorrect double property ranges |
| Fixed | 9346 | Chart Trader | Pending order modification persists when switching windows |
| Fixed | 9347 | Chart | Day/Week/Month/Year chart lookback defaults were  incorrect |
| Fixed | 9350 | NinjaScript | Export failed when excluded drawing tool has compilation errors |
| Fixed | 9352 | NinjaScript | Using a Strokes .BrushDX property caused the platform to crash |
| Fixed | 9354 | Strategy Analyzer | Strategy Analyzer "Chart" Drawing object showed extra z-order after second run |
| Fixed | 9355 | NinjaScript Editor, UI | NinjaScript incorrect 'auto focused' when maximized |
| Fixed | 9356 | Playback | Playback start date generated error too soon when trying to type in a new year |
| Fixed | 9357 | Data Grids | Sorting Partial Fills in Executions Tab did  not sort as expected |
| Fixed | 9358 | Strategy Analyzer | Errors were generated when running SampleCumProfit on 64-bit |
| Fixed | 9359 | DrawingTool | Strategy Global Drawing Object Were  Not Removing After Strategy Removed |
| Fixed | 9360 | Playback | Chart "Days to load" was changing after connecting to Playback |
| Fixed | 9361 | Backup & Restore, NinjaScript | Unable to import NinjaScript when image file exists |
| Fixed | 9362 | Chart, DrawingTool | Drawing objects scaled inconsistently on multi series charts |
| Fixed | 9364 | Strategy | Exception on startup when strategy on chart |
| Fixed | 9378 | DrawingTool | Draw.RegionHighlight..() not using templates "AreaColor" |
| Fixed | 9400 | Adapter | Changed "Run time pop up handling" to default on new connection to prevent TWS stealing focus when Global Config window open. |
| Fixed | 6135 | Alerts | Performance - Alerts / Alert Log |

|  |  |  |  |
| --- | --- | --- | --- |
| Changed | 9340 | Trading Technologies | Removed TT Adapter |
| **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" \l "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" \l "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" \l "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:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Performance Metrics](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) >  **OnCopyTo()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/onaddtrade.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/onmergeperformancemetric.htm) |

**Definition**

Called as the values of a trade metric are saved.

**Syntax**

protected override void OnCopyTo(PerformanceMetricBase target)  
{  
   
}

**Examples**

| ns | |
| --- | --- |
| protected override void OnCopyTo(PerformanceMetricBase target) {   // You need to cast, in order to access the right type   SampleCumProfit targetMetrics = (target as SampleCumProfit);     if (targetMetrics != null)     Array.Copy(Values, targetMetrics.Values, Values.Length); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [ISeries<T>](https://ninjatrader.com/es/support/helpGuides/nt8/iseriest.htm) >  **Series<T>** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/iseriest.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/iseriest.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/reset.htm) |

**Definition**

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

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

**Parameters**

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

**Methods and Properties**

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

**Creating Series<T> Objects**

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

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

To create a Series<T> object:

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

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

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

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

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

**Setting Values**

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

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

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

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

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

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

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

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

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

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

| ns **Using a custom Series object as indicator input** |
| --- |
| protected override void OnBarUpdate() {   // Calculate the range of the current bar and set the value     myDoubleSeries[0] = (High[0] - Low[0]);       // Print the current 10 period SMA of range     Print("Value is " + SMA(myDoubleSeries, 10)[0]);         } |