|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) >  **ISeries<T>** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/url.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/seriest.htm) |

**Definition**

ISeries<T> is an interface that is implemented by all NinjaScript classes that manage historical data as an ISeries<double> (Open, High, Low, Close, etc), used for indicator input, and other object data.  Please see the help guide article on [Working with Price Series](https://ninjatrader.com/es/support/helpGuides/nt8/working_with_price_series.htm) for a basic overview on how to access this information.

**Types of ISeries**

|  |  |
| --- | --- |
| [Series<T>](https://ninjatrader.com/es/support/helpGuides/nt8/seriest.htm) | Represents a generic custom data structure for custom development |
| [PriceSeries](https://ninjatrader.com/es/support/helpGuides/nt8/priceseries.htm) | Historical price data structured as an ISeries<double> interface (Close[0], High[0], Low[0], etc) |
| [TimeSeries](https://ninjatrader.com/es/support/helpGuides/nt8/timeseries.htm) | Historical time stamps structured as an ISeries<DateTime> interface (Time[0]) |
| [VolumeSeries](https://ninjatrader.com/es/support/helpGuides/nt8/volumeseries.htm) | Historical volume data structured as an ISeries<double> interface (Volume[0]) |

**Methods and Properties**

|  |  |
| --- | --- |
| [GetValueAt()](https://ninjatrader.com/es/support/helpGuides/nt8/getvalueat.htm) | Returns the underlying input value at a specified bar index value. |
| [IsValidDataPoint()](https://ninjatrader.com/es/support/helpGuides/nt8/isvaliddatapoint.htm) | Indicates if the specified input is set at a barsAgo value relative to the current bar. |
| [IsValidDataPointAt()](https://ninjatrader.com/es/support/helpGuides/nt8/isvaliddatapointat.htm) | Indicates if the specified input is set at a specified bar index value. |
| [Count](https://ninjatrader.com/es/support/helpGuides/nt8/iseries_count.htm) | Return the number total number of values in the ISeries array |

|  |
| --- |
| **Tips**: (see examples below)  1.By specifying a parameter of type ISeries<double>, you can then pass in an array of closing prices, an indicator, or a user defined data series.  2.When working with ISeries<double> objects in your code you may come across situations where you are not sure if the value being accessed is a valid value or just a "placeholder" value. To check if you are using valid values for your logic calculations that have been explicitly set, please use .IsValidDataPoint(int *barsAgo*)to check. |

**Examples**

| ns | **Using ISeries as a method parameter** |
| --- | --- |
|  | //create custom a method named DoubleTheValue that accepts any object that implements // the ISeries<double> interface as a parameter private double DoubleTheValue(ISeries<double> priceData) {     return priceData[0] \* 2; }   protected override void OnBarUpdate() {   // This custom method is then used twice,   //the first time passing in an array of closing prices     Print(DoubleTheValue(Close));   //and the second time passing in a 20 period simple moving average.     Print(DoubleTheValue(SMA(20))); } |

| ns | **Checking ISeries value before accessing** |
| --- | --- |
|  | protected override void OnBarUpdate() {     // Only set our plot if the input is a valid value     if (Input.IsValidDataPoint(0))         Plot0[0] = Input[0]; } |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [ISeries<T>](https://ninjatrader.com/es/support/helpGuides/nt8/iseriest.htm) > [PriceSeries<double>](https://ninjatrader.com/es/support/helpGuides/nt8/priceseries.htm) >  **Input** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/highs.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/priceseries.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/inputs.htm) |

**Definition**

The main historical data input. If implemented in the NinjaScript object, it allows for more flexibility as non bars based series such as plot series could be passed in and drive the calculation outcomes - an example would be a custom moving average that should have the ability to operate on another moving average (i.e. the SMA) as input series.

**Property Value**

An ISeries<double> type object that implements the Series<double> interface. Accessing this property via an index value [int *barsAgo*] returns a double value representing the price of the referenced bar.

**Syntax**

Input  
Input[int *barsAgo*]

**Examples**

| ns |
| --- |
| // Prints the the current value of input Print(Input[0].ToString()); |

| ns |
| --- |
| // Prints the the current type of input passed to the object, so we can detect if we're working on a price based series such as OHLCV or a derivative such as an SMA indicator if (Input is PriceSeries) Print("Price Series Input"); if (Input is Indicator) Print("Indicator Input"); |

|  |  |
| --- | --- |
| **Tip**:When working with multi-series indicators, Input is not guaranteed to reference the primary [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm). Please be mindful as to when you access Input[0] as you will only be able to do so after the contextual BarsInProgress has bars. To check to ensure BarsInProgress has some bars you can use [CurrentBars](https://ninjatrader.com/es/support/helpGuides/nt8/currentbars.htm) to check. | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **BarsSinceNewTradingDay** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getask.htm) |

**Definition**

Returns the number of bars elapsed since the start of the trading day relative to the current bar processing.

**Property Value**

An int value representing the number of bars elapsed.  This property cannot be set.

**Syntax**  
Bars.BarsSinceNewTradingDay

**Examples**

| ns | |
| --- | --- |
| // Only process strategy logic after five bars have posted since the start of the trading day protected override void OnBarUpdate() {   if (Bars.BarsSinceNewTradingDay >= 5)   {     //Strategy logic here   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **GetAsk()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/barssincenewtradingday.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getbar.htm) |

**Definition**

Returns the ask price value at a selected absolute bar index value.

|  |
| --- |
| **Note**: This method does **NOT** return the current real-time asking price, but rather the historical/real-time asking price at the desired index.  For obtaining the current real-time asking price, please use [GetCurrentAsk](https://ninjatrader.com/es/support/helpGuides/nt8/getcurrentask.htm)(). |

**Method Return Value**

A double value that represents the asking price at the desired bar index.

**Syntax**

Bars.GetAsk(int index)

**Parameters**

|  |  |
| --- | --- |
| index | The absolute bar index value used |

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {   // If the Highs of the two most recent bars are falling, place a long stop market order   // at the Ask price   if (High[0] < High[1] && High[1] < High[2])   {       EnterLongStopMarket(Bars.GetAsk(CurrentBar));   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **GetBar()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getask.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getbid.htm) |

**Definition**

Returns the first bar that matches the time stamp of the "time" parameter provided.

|  |
| --- |
| **Note**:  If the time parameter provided is older than the first bar in the series, a bar index of 0 is returned. If the time stamp is newer than the last bar in the series, the last absolute bar index is returned. |

**Method Return Value**

An int value representing an absolute bar index value.

**Syntax**  
Bars.GetBar(DateTime time)

**Parameters**

|  |  |
| --- | --- |
| time | Time stamp to be converted to an absolute bar index |

**Examples**

| ns | |
| --- | --- |
| // Check that its past 9:45 AM if (ToTime(Time[0]) >= ToTime(9, 45, 00)) {   // Calculate the bars ago value for the 9 AM bar for the current day   int barsAgo = CurrentBar - Bars.GetBar(new DateTime(2006, 12, 18, 9, 0, 0));     // Print out the 9 AM bar closing price   Print("The close price on the 9 AM bar was: " + Close[barsAgo].ToString()); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **GetBid()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getbar.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getclose.htm) |

**Definition**

Returns the bid price value at a selected absolute bar index value.

|  |
| --- |
| **Note**: This method does **NOT** return the current real-time bid price, but rather the historical/real-time bid price at the desired index.  For obtaining the current real-time bid price, please use [GetCurrentBid()](https://ninjatrader.com/es/support/helpGuides/nt8/getcurrentbid.htm). |

**Method Return Value**

A double value that represents the biding price at the desired bar index.

**Syntax**

Bars.GetBid(int index)

**Parameters**

|  |  |
| --- | --- |
| index | The absolute bar index value used |

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {   // If the Highs of the two most recent bars are falling, place a long stop market order   // at the Ask price   if (Low[0] > Low[1] && Low[1] < Low[2])   {     EnterShortStopMarket(Bars.GetBid(CurrentBar));   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **GetClose()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getbid.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getdaybar.htm) |

**Definition**

Returns the closing price at the current bar index value.

**Method Return Value**

A double value that represents the close price at the desired bar index.

**Syntax**

Bars.GetClose(int index)

**Parameters**

|  |  |
| --- | --- |
| index | An int representing an absolute bar index value |

**Examples**

| ns | |
| --- | --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   base.OnRender(chartControl, chartScale);   // loop through only the rendered bars on the chart   for(int barIndex = ChartBars.FromIndex; barIndex <= ChartBars.ToIndex; barIndex++)   {     // get the close price at the selected bar index value     double closePrice = Bars.GetClose(barIndex);     Print("Bar #" + barIndex + " closing price is " + closePrice);   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **GetDayBar()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getclose.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/gethigh.htm) |

**Definition**

Returns a virtual historical Bar object that represents a trading day whose properties for open, high, low, close, time and volume can be accessed.

|  |
| --- |
| **Notes:**  1.The bar object returned is a "virtual bar" built from the underlying bar series and its configured session.  Since the bar object is virtual, its property values are calculated based on session definitions contained in the trading day only. The returned bar object does **NOT** necessarily represent the actual day.  For accessing a true "Daily" bar, please see use [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm)and use the BarsPeriodType.Day as the bars period.  2.GetDayBar() should **ONLY** be used for accessing prior trading day data. To access current trading day data, use the [CurrentDayOHL()](https://ninjatrader.com/es/support/helpGuides/nt8/current_day_ohl.htm) method. |

**Method Return Value**

A virtual bar object representing the current configured session. Otherwise null if there is insufficient intraday data

**Syntax**  
The properties below return double values:

Bars.GetDayBar(int tradingDaysBack).Open  
Bars.GetDayBar(int tradingDaysBack).High  
Bars.GetDayBar(int tradingDaysBack).Low  
Bars.GetDayBar(int tradingDaysBack).Close

The property below returns a [DateTime](http://msdn.microsoft.com/en-us/library/system.datetime.aspx" \t "_blank) structure:

Bars.GetDayBar(int tradingDaysBack).Time

The property below returns an int value:

Bars.GetDayBar(int tradingDaysBack).Volume

|  |
| --- |
| **Warning**:  You must check for a null reference to ensure there is sufficient intraday data to build a trading day bar. |

**Parameters**

|  |  |
| --- | --- |
| tradingDaysBack | An int representing the number of the trading day to get OHLCV and time information from |

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {   // Check to ensure that sufficient intraday data was supplied   if(Bars.GetDayBar(1) != null)     Print("The prior trading day's close is: " + Bars.GetDayBar(1).Close); } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **GetDayBar()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getclose.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/gethigh.htm) |

**Definition**

Returns a virtual historical Bar object that represents a trading day whose properties for open, high, low, close, time and volume can be accessed.

|  |
| --- |
| **Notes:**  1.The bar object returned is a "virtual bar" built from the underlying bar series and its configured session.  Since the bar object is virtual, its property values are calculated based on session definitions contained in the trading day only. The returned bar object does **NOT** necessarily represent the actual day.  For accessing a true "Daily" bar, please see use [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm)and use the BarsPeriodType.Day as the bars period.  2.GetDayBar() should **ONLY** be used for accessing prior trading day data. To access current trading day data, use the [CurrentDayOHL()](https://ninjatrader.com/es/support/helpGuides/nt8/current_day_ohl.htm) method. |

**Method Return Value**

A virtual bar object representing the current configured session. Otherwise null if there is insufficient intraday data

**Syntax**  
The properties below return double values:

Bars.GetDayBar(int tradingDaysBack).Open  
Bars.GetDayBar(int tradingDaysBack).High  
Bars.GetDayBar(int tradingDaysBack).Low  
Bars.GetDayBar(int tradingDaysBack).Close

The property below returns a [DateTime](http://msdn.microsoft.com/en-us/library/system.datetime.aspx" \t "_blank) structure:

Bars.GetDayBar(int tradingDaysBack).Time

The property below returns an int value:

Bars.GetDayBar(int tradingDaysBack).Volume

|  |
| --- |
| **Warning**:  You must check for a null reference to ensure there is sufficient intraday data to build a trading day bar. |

**Parameters**

|  |  |
| --- | --- |
| tradingDaysBack | An int representing the number of the trading day to get OHLCV and time information from |

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {   // Check to ensure that sufficient intraday data was supplied   if(Bars.GetDayBar(1) != null)     Print("The prior trading day's close is: " + Bars.GetDayBar(1).Close); } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **GetHigh()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getdaybar.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getlow.htm) |

**Definition**

Returns the high price at the selected bar index value.

**Method Return Value**

A double value that represents the high price at the desired bar index.

**Syntax**

Bars.GetHigh(int index)

**Parameters**

|  |  |
| --- | --- |
| index | An int representing an absolute bar index value |

**Examples**

| ns | |
| --- | --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   base.OnRender(chartControl, chartScale);   // loop through only the rendered bars on the chart   for(int barIndex = ChartBars.FromIndex; barIndex <= ChartBars.ToIndex; barIndex++)   {     // get the high price at the selected bar index value     double highPrice = Bars.GetHigh(barIndex);     Print("Bar #" + barIndex + " high price is " + highPrice);   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **GetLow()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/gethigh.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getopen.htm) |

**Definition**

Returns the low price at the selected bar index value.

**Method Return Value**

A double value that represents the low price at the desired bar index.

**Syntax**

Bars.GetLow(int index)

**Parameters**

|  |  |
| --- | --- |
| index | An int representing an absolute bar index value |

**Examples**

| ns | |
| --- | --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   base.OnRender(chartControl, chartScale);   // loop through only the rendered bars on the chart   for(int barIndex = ChartBars.FromIndex; barIndex <= ChartBars.ToIndex; barIndex++)   {     // get the low price at the selected bar index value     double lowPrice = Bars.GetLow(barIndex);     Print("Bar #" + barIndex + " low price is " + lowPrice);   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **GetOpen()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getlow.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getsessionendtime.htm) |

**Definition**

Returns the open price at the selected bar index value.

**Method Return Value**

A double value that represents the open price at the desired bar index.

**Syntax**

Bars.GetOpen(int index)

**Parameters**

|  |  |
| --- | --- |
| index | An int representing an absolute bar index value |

**Examples**

| ns | |
| --- | --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   base.OnRender(chartControl, chartScale);   // loop through only the rendered bars on the chart   for(int barIndex = ChartBars.FromIndex; barIndex <= ChartBars.ToIndex; barIndex++)   {     // get the open price at the selected bar index value     double openPrice = Bars.GetOpen(barIndex);     Print("Bar #" + barIndex + " open price is " + openPrice);   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **GetTime()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getsessionendtime.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/getvolume.htm) |

**Definition**

Returns the time stamp at the current bar index value.

|  |
| --- |
| **Note**: This method will return what is displayed in the chart's data box.  For formatting purposes, the value returned is **NOT**guaranteed be equal to the [TimeSeries](https://ninjatrader.com/es/support/helpGuides/nt8/timeseries.htm) value.  If you are using daily bars and need the session end time, you should use [Bars.GetSessionEndTime()](https://ninjatrader.com/es/support/helpGuides/nt8/getsessionendtime.htm) instead. |

**Method Return Value**

A DateTime structure that represents the time stamp at the desired bar index.

**Syntax**

Bars.GetTime(int index)

**Parameters**

|  |  |
| --- | --- |
| index | An int representing an absolute bar index value |

**Examples**

| ns | |
| --- | --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   base.OnRender(chartControl, chartScale);   // loop through only the rendered bars on the chart   for(int barIndex = ChartBars.FromIndex; barIndex <= ChartBars.ToIndex; barIndex++)   {     // get the time stamp at the selected bar index value     DateTime timeValue = Bars.GetTime(barIndex);     Print("Bar #" + barIndex + " time stamp is " + timeValue);   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **GetVolume()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/gettime.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isfirstbarofsession.htm) |

**Definition**

Returns the volume at the selected bar index value.

**Method Return Value**

A long value represents the volume at the desired bar index.

**Syntax**

Bars.GetVolume(int index)

**Parameters**

|  |  |
| --- | --- |
| index | An int representing an absolute bar index value |

**Examples**

| ns | |
| --- | --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   base.OnRender(chartControl, chartScale);   // loop through all the rendered bars on the chart   for(int barIndex = ChartBars.FromIndex; barIndex <= ChartBars.ToIndex; barIndex++)   {     // get the volume value at the selected bar index value     long volumeValue = Bars.GetVolume(barIndex);     Print("Bar #" + barIndex + " volume value is " + volumeValue);   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **IsFirstBarOfSession** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getvolume.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isfirstbarofsessionbyindex.htm) |

**Definition**

Indicates if the current bar processing is the first bar updated in a trading session.

|  |
| --- |
| **Note**:  This property always returns **true** on the very first bar processed (i.e., CurrentBar == 0).  The represented time of the bar will **NOT** necessarily be equal to the trading hours start time (e.g., if you request 50 1-minute bars at 11:50:00 AM, the first bar processed of the session would be 11:00:00 AM).  Loading a data series based on "dates" (Days or custom range) ensures that the first bar processed matches hours defined by the session template. |

**Property Value**

This property returns **true** if the bar is the first processed in a session; otherwise, **false**.  This property is read-only.

|  |
| --- |
| **Warning**:   This property will always return **false** on non-intraday bar periods (e.g., Day, Month, etc).  For checking for new non-intraday bar updates, please see [IsFirstTickOfBar](https://ninjatrader.com/es/support/helpGuides/nt8/isfirsttickofbar.htm) |

**Syntax**  
Bars.IsFirstBarOfSession

|  |
| --- |
| **Tip**:  For checking at a specified bar index, please see [IsFirstBarOfSessionByIndex()](https://ninjatrader.com/es/support/helpGuides/nt8/isfirstbarofsessionbyindex.htm) |

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {   // Print the current bar number of the first bar processed for each session on a chart   if (Bars.IsFirstBarOfSession)     Print(string.Format("Bar number {0} was the first bar processed of the session at {1}.", CurrentBar, Time[0])); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **IsFirstBarOfSessionByIndex()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isfirstbarofsession.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/islastbarofsession.htm) |

**Definition**

Indicates if the selected bar index value is the first bar of a trading session.

**Property Value**

This property returns **true** if the bar is the first bar of a session; otherwise, **false**.  This property is read-only.

**Syntax**  
Bars.IsFirstBarOfSessionByIndex(int index)

|  |
| --- |
| **Warning**:   This property will always return **false** on non-intraday bar periods (e.g., Day, Month, etc) |

**Parameters**

|  |  |
| --- | --- |
| index | An int representing an absolute bar index value |

**Examples**

| ns | |
| --- | --- |
| protected override void OnRender(ChartControl chartControl, ChartScale chartScale) {   base.OnRender(chartControl, chartScale);   //  loop through only the rendered bars on the chart   for(int barIndex = ChartBars.FromIndex; barIndex <= ChartBars.ToIndex; barIndex++)   {     // check if the rendered bar is the first bar of the trading session     if (Bars.IsFirstBarOfSessionByIndex(barIndex))     {         DateTime slotTimeAtBarIndex = chartControl.GetTimeBySlotIndex(barIndex);         Print(string.Format("Bar index {0} was the first bar of the session at slot time {1}.", barIndex, slotTimeAtBarIndex));     }   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **IsLastBarOfSession** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isfirstbarofsessionbyindex.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isresetonnewtradingday.htm) |

**Definition**

Indicates if the current bar processing is the last bar updated in a trading session.

|  |
| --- |
| **Notes:**  •This property will always return **false** on non-intraday bar periods (e.g., Day, Month, etc.)  •When running **Calculate.OnEachTick** / **OnPriceChange**, this property will always return **true** on the most current real-time bar since it is the last bar that is updating in the trading session.  If you need to find a bar which coincides with the session end time, please use the [SessionIterator.ActualSessionEnd](https://ninjatrader.com/es/support/helpGuides/nt8/actualsessionend.htm). |

**Property Value**

This property returns **true** if the bar is the last processed in a session; otherwise, **false**.  This property is read-only.

**Syntax**  
Bars.IsLastBarOfSession

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {   // Print the current bar number of the first bar processed for each session on a chart   if(Bars.IsLastBarOfSession)     Print(string.Format("Bar number {0} was the last bar processed of the session at {1}.", CurrentBar, Time[0])); } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **IsResetOnNewTradingDay** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/islastbarofsession.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/istickreplay.htm) |

**Definition**

Indicates if the bars series is using the [Break EOD](https://ninjatrader.com/es/support/helpGuides/nt8/break_at_eod.htm) data series property.

**Property Value**

This property returns **true** if the bars series should reset on a new trading day; otherwise, **false**.  This property is read-only.

**Syntax**  
Bars.IsResetOnNewTradingDay

|  |
| --- |
| **Tip**: This property can be helpful in determine on how to amend new bar data when working with a [BarType](https://ninjatrader.com/es/support/helpGuides/nt8/bars_type.htm) |

**Examples**

| ns | |
| --- | --- |
| protected override void OnDataPoint(Bars bars, double open, double high, double low, double close, DateTime time, long volume, bool isBar, double bid, double ask) {   // create a session iterator to keep track of session related information   if(SessionIterator == null)     SessionIterator = new SessionIterator(bars);    // determine if the bars are in a new session   bool isNewSession = SessionIterator.IsNewSession(time, isBar);    if(isNewSession)     SessionIterator.GetNextSession(time, isBar);     // If bars are using "Break end of day", add a new bar for next session   if(bars.IsResetOnNewTradingDay && isNewSession))     AddBar(bars, open, high, low, close, time, volume);   else   {     // do something with existing bar values   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **IsTickReplay** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isresetonnewtradingday.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/percentcomplete.htm) |

**Definition**

Indicates if the bar series is using the [Tick Replay](https://ninjatrader.com/es/support/helpGuides/nt8/developing_for__tick_replay.htm) data series property.

**Property Value**

This property returns **true** if the bar series is using tick replay; otherwise, **false**.  This property is read-only.

**Syntax**  
Bars.IsTickReplay

|  |
| --- |
| **Warning**: A **Tick Replay** indicator or strategy **CANNOT**use a **MarketDataType.Ask** or **MarketDataType.Bid**series.  Please see [Developing for Tick Replay](https://ninjatrader.com/es/support/helpGuides/nt8/developing_for__tick_replay.htm) for more information. |

**Examples**

| ns | |
| --- | --- |
| private double askPrice; protected override void OnMarketData(MarketDataEventArgs marketDataUpdate) {   if(Bars.IsTickReplay)   {     // if using tick replay, get the current ask price associated with the tick     askPrice = marketDataUpdate.Ask;   }   else // otherwise, get the real-time market data price during MarketDataType.Ask event     askPrice = marketDataUpdate.MarketDataType == MarketDataType.Ask ? marketDataUpdate.Price : double.MinValue;    // only print if a value is set   if(askPrice != double.MinValue)   {     Print("ask price: " + askPrice);   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **PercentComplete** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/istickreplay.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/tickcount.htm) |

**Definition**

Returns a value indicating the percentage complete of the real-time bar processing.

|  |
| --- |
| **Notes**:  1.Since a historical bar is complete, values during **State.Historical** should be ignored (also the case with [TickReplay](https://ninjatrader.com/es/support/helpGuides/nt8/developing_for__tick_replay.htm) bars)  2.Some [BarsTypes](https://ninjatrader.com/es/support/helpGuides/nt8/bars_type.htm) may not be compatible with the **PercentComplete** property. In these cases, a value of 0 always returns (e.g.,  Range, Renko, Point & Figure, Kagi, LineBreak, and some other 3rd party bars types) |

**Property Value**

A double value representing a percent e.g. a value of .5 indicates the bar was at 50%.  This property is read-only.

**Syntax**  
Bars.PercentComplete

|  |
| --- |
| **Tip**:  If you are developing a custom **BarsType**, please use the [GetPercentComplete()](https://ninjatrader.com/es/support/helpGuides/nt8/getpercentcomplete.htm) method used to calculate the value returned by **PercentComplete** |

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {   if(State == State.Realtime)   {     Draw.TextFixed(this, "barstatus", Bars.PercentComplete.ToString("P2"), TextPosition.BottomRight);   } } | |
| **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) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **TickCount** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/percentcomplete.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/tochartstring.htm) |

**Definition**

Returns the total number of ticks of the current bar processing.

|  |
| --- |
| **Note**:  For historical usage, you must use **Calculate.OnEachTick** with [TickReplay](https://ninjatrader.com/es/support/helpGuides/nt8/developing_for__tick_replay.htm) enabled; otherwise a value of 1 will returned. |

**Property Value**

A long value that represents the total number of ticks of the current bar.

**Syntax**

Bars.TickCount

**Examples**

| ns | |
| --- | --- |
| // Prints the tick count to the output window Print("The tick count of the current bar is " + Bars.TickCount.ToString()); | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Bars](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) >  **ToChartString()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/tickcount.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/bars.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/chart.htm) |

**Definition**

Returns the bars series as a formatted string, including the [Instrument.FullName](https://ninjatrader.com/es/support/helpGuides/nt8/instrument_fullname.htm), [BarsPeriod](https://ninjatrader.com/es/support/helpGuides/nt8/barsperiod.htm) Value, and BarsPeriodType name.

|  |
| --- |
| **Note**:  To obtain a return value which matches the user configured [ChartBars Label property](https://ninjatrader.com/es/support/helpGuides/nt8/chartbars_properties.htm), please see the [ChartBars.ToChartString()](https://ninjatrader.com/es/support/helpGuides/nt8/chartbars_tochartstring().htm) method |

**Syntax**

Bars.ToChartString()

**Return Value**

A string value that represents the bars series

**Parameters**

This method does not accept any parameters

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {   // print the chart string on start up   if(CurrentBar == 0)     Print(Bars.ToChartString()); // ES 09-15 (60 Minute)       } |