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

**Definition**

The optimization parameters selected for the optimization run. (e.g. user parameters or Data Series)

**Property Value**

A bool value.

**Syntax**

Strategies[0].OptimizationParameters

**Examples**

| ns | |
| --- | --- |
| protected override void OnOptimize()  {      // If there are no optimization parameters to optimize, return      if (Strategies[0].OptimizationParameters.Count == 0)          return;        // Do something with the optimization parameter      Parameter parameter = Strategies[0].OptimizationParameters[0];  } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Educational Resources](https://ninjatrader.com/es/support/helpGuides/nt8/educational_resources.htm) > [Tips](https://ninjatrader.com/es/support/helpGuides/nt8/tips.htm) >  **Parameter sequencing** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/order_types.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tips.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/referencing_the_correct_bar.htm) |

Indicator and strategy parameters (user defined inputs) will always be displayed in an order that the user specifies in the NinjaScript file.

In the NinjaScript Editor, expand the "Properties" region of your code where all of your parameters are defined. In this example, this will be our Properties section:

| ns |
| --- |
| [Range(1, int.MaxValue), NinjaScriptProperty] [Display(ResourceType = typeof(Custom.Resource), Name = "Fast", GroupName = "NinjaScriptStrategyParameters", Order = 0)] public int Fast { get; set; } [Range(1, int.MaxValue), NinjaScriptProperty] [Display(ResourceType = typeof(Custom.Resource), Name = "Slow", GroupName = "NinjaScriptStrategyParameters", Order = 1)] public int Slow { get; set; } |

In this case, the Fast parameter will show up as the first parameter with the Slow parameter showing as the second.

To switch the order around, we could modify Order. If we change Slow's Order to 0 and Fast's Order to 1 as shown below ...

| ns |
| --- |
| [Range(1, int.MaxValue), NinjaScriptProperty] [Display(ResourceType = typeof(Custom.Resource), Name = "Fast", GroupName = "NinjaScriptStrategyParameters", Order = 1)] public int Fast { get; set; } [Range(1, int.MaxValue), NinjaScriptProperty] [Display(ResourceType = typeof(Custom.Resource), Name = "Slow", GroupName = "NinjaScriptStrategyParameters", Order = 0)] public int Slow { get; set; } |

... the Slow property will show first and the Fast property second.

|  |  |
| --- | --- |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Educational Resources](https://ninjatrader.com/es/support/helpGuides/nt8/educational_resources.htm) > [Tips](https://ninjatrader.com/es/support/helpGuides/nt8/tips.htm) >  **Creating User Defined Input Parameters** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/checking_for_null_references.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/tips.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/debugging_your_ninjascript_cod.htm) |

You can create user defined input parameters for both NinjaScript Indicators and Strategies. Although user defined input parameters can be specified as part of the initial set up of NinjaScript Indicator or Strategies using the Wizard you may have a requirement to add new parameters at a later point in your development process. To create these parameters you will need to edit your NinjaScript code and follow these steps.

1.Open your NinjaScript file

2.Inside of the if (State == State.SetDefaults) statement, assign a value to the variable for your parameter

| ns |
| --- |
| Period = 5; |

|  |
| --- |
| **Note**: This is also where you set the default value for your parameter. |

3.Scroll down to the bottom of the editor and expand the minimized "Properties" section by clicking on the + sign on the left.

4.Use the following template code for each parameter you wish to create. Please note that the type (int, double, etc) will differ depending on what type of variable you wish to create

| ns |
| --- |
| [Range(1, int.MaxValue)] [NinjaScriptProperty] [Display(Name="Period", Description="Numbers of bars used for calculations", Order=1, GroupName="Parameters")] public int Period { get; set; } |

5.To specify lower and upper bounds, you would modify [Range(1, int.MaxValue)]. For example:

| ns |
| --- |
| *// No upper bound, lower bound of 1* [Range(1, int.MaxValue)] *// No lower bound, upper bound of 100* [Range(int.MinValue, 100)] *// No lower or upper bound* [Range(int.MinValue, int.MaxValue)] |

6.Use the "Description" field to provide a brief description of what the parameter does.

7.Pay attention to this line as the object type will vary depending on the type of parameter you wish to make:

| ns |
| --- |
| public int Period |

8.Now, wherever in your code you want to call the user-definable parameter, just use "Period".

| ns | |
| --- | --- |
| if (SMA(Period)[0] > SMA(Period)[1])     *// Do something* | |
| **Navigation:**  »No topics above this level«  **Condition Builder** | | [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/sharpdx_directwrite_textlayout.htm) |

The **Condition Builder** is a very powerful feature that allows you to define complex conditions for your automated trading systems without having to know how to program.

tog_minus        [Understanding the Condition Builder](javascript:HMToggle('toggle','UnderstandingTheConditionBuilder','UnderstandingTheConditionBuilder_ICON'))

|  |
| --- |
| **Condition Builder**  Most if not all automated trading system code wizards are limited in scope in that they provide canned predefined expressions and only allow you to change a few parameters on those expressions. The NinjaTrader **Condition Builder** is advanced in that you can develop powerful expressions without limitations. Due to its power and flexibility, it is extremely important that you read through and understand its capabilities.    The **Condition Builder**is also a very powerful aid for those of you learning NinjaScript or learning how to program. You can build your conditions within the **Condition Builder** and instantly see NinjaScript code generated by having the [NinjaScript Editor](https://ninjatrader.com/es/support/helpGuides/nt8/editor.htm) open (by pressing the **View Code...** button in the Builder screen).    The **Condition Builder** can be accessed via the [Conditions and Actions](https://ninjatrader.com/es/support/helpGuides/nt8/builder_screens.htm) screen in the NinjaTrader Strategy Builder.    **Basic Operation**  The general concept of the **Condition Builder** is to generate a Boolean expression also known as comparison expressions or conditional expressions. What does that mean? It is simply an expression that results in a value of either TRUE or FALSE. For example, the expression    **2 < 7 (2 is less than 7)**    is a Boolean expression because the result is TRUE. All expressions that contain relational operators are Boolean. Boolean expressions or "Conditions" as they are known in NinjaTrader is used to determine when to take a specified action such as submitting an order or drawing on the chart.    Looking at the image below, you can instantly see that the **Condition Builder** is set up like a Boolean expression. Select an item from the left window (1), compare it to a selected item in the right window (1) and then select the relational operator (2).    Strategy_Builder_CB1    1. Available items such as indicators, price data, etc. to use for the comparison  2. List of relational operators (the relational operator will *only list the comparisons applicable* to your comparison item choices *and only after both sides* have been selected) |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?condition_builder2.htm#UnderstandingTheConditionBuilder)

tog_minus        [How to make price data comparisons](javascript:HMToggle('toggle','HowToMakePriceDataComparisons','HowToMakePriceDataComparisons_ICON'))

|  |
| --- |
| **Price Data Comparisons**  You can compare a bar's price data such as checking for a higher close. The following is an example and represents one of many possible combinations.    1. Expand the **Price** category on the left side and select the **Close**.  2. Expand the **Price** category on the right side and select the **Close**.  3. Select the **greater** relational operator  4. Set the **Bars ago** parameter to a value of "1"    Strategy_Builder_CB2  Once the **OK** button is pressed, a condition is created that would translate to the following:    **"Current closing price is greater than the closing price of 1 bar ago"** |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?condition_builder2.htm#HowToMakePriceDataComparisons)

tog_minus        [How to offset an item value](javascript:HMToggle('toggle','HowToOffsetAnItemValue','HowToOffsetAnItemValue_ICON'))

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Offsetting an Item Value**  You can offset the value of most items available in the **Condition Builder**. An offset is a value that is added, subtracted, multiplied or divided from / into the actual item's value. When an item is selected such as an indicator or price data, the **Offset** and **Offset** **type**parameters become visible in the window directly below the item selected. This is shown as numbers 5 and 6 in the image below.    **Offset type** can be set to:     |  |  | | --- | --- | | Arithmetic | Offsets by an arithmetic equation you can setup by the absolute value and the arithmetic offset operator to the left (+ - \* /) | | Pips | Offsets by the specified amount of pips | | Percent | Offsets a percentage value of the item's value. A value of 1 is equal to 100% where a value of 0.1 is equal to 10%. | | Ticks | Offsets by the specified amount of ticks |     Once the **Offset type** is selected, you must set the value **Offset**. In addition to the example below, you can see the "*Checking for Volume Expansion*" section below for another example that uses the **Percent** **Offset type**.    The following is an example and represents one of many possible combinations:    1. Expand the **Price** category and select the **Close**  2. Expand the **Price** category and select the **High**  3. Select the **greater** relational operator  4. Set the **Bars ago** parameter to a value of "1"  5. Set the **Offset type** parameter to **Ticks**  6. Set the **Offset** parameter to a value of "1"    Strategy_Builder_CB3    Once the **OK** button is pressed, a condition is created that would translate to the following:    **"Current closing price is greater than the high price of 1 bar ago + 1 tick"** |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?condition_builder2.htm#HowToOffsetAnItemValue)

tog_minus        [How to make indicator to value comparisons](javascript:HMToggle('toggle','HowToMakeIndicatorToValueComparisons','HowToMakeIndicatorToValueComparisons_ICON'))

|  |
| --- |
| **Indicator to Value comparisons**  You can compare an indicator's value to a numeric value. This can come in handy if you wanted to check if ADX is over a value of 30 (trending) or if Stochastics is under a value of 20 (oversold) or any other conditions you can think of.    The following is an example and represents one of many possible combinations:    1. Expand the **Indicator** category and select the **ADX** indicator  2. Set the parameters of the indicator, for our example with the default values no changes are needed  3. Expand the **Misc**category and select **Numeric value**  4. Select the **greater** relational operator  5. Enter the numeric value you want to compare the indicator to (30 in our example)    Strategy_Builder_CB4  Once the **OK** button is pressed, a condition is created that would translate to the following:    **"Current value of a 14 period ADX is greater than 30"** |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?condition_builder2.htm#HowToMakeIndicatorToValueComparisons)

tog_minus        [How to compare plot values of multi-plot indicators](javascript:HMToggle('toggle','HowToComparePlotValuesOfMultiplotIndicators','HowToComparePlotValuesOfMultiplotIndicators_ICON'))

|  |
| --- |
| **Comparing Plot Values of Multi-Plot indicators**  You can compare plots in the same indicator or select any individual plot within an indicator to create a condition.    The following is an example and represents one of many possible combinations:    1. Expand the **Indicator** category and select the **Stochastics** indicator  2. Set the indicator input parameters and select the **K** plot (green arrow)  3. Expand the **Indicator** category and select the **Stochastics** indicator  4. Select the **greater** relational operator  5. Set the indicator input parameters and select the **D** plot (green arrow)    Strategy_Builder_CB5  Once the **OK** button is pressed, a condition is created that would translate to the following:    **"Current K plot value of a Stochastics indicator is greater than the current D plot value of the same Stochastics indicator"** |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?condition_builder2.htm#HowToComparePlotValuesOfMultiplotIndicators)

tog_minus        [How to use user inputs & variables](javascript:HMToggle('toggle','HowToUseUserDefinedInputsAndVariables','HowToUseUserDefinedInputsAndVariables_ICON'))

|  |
| --- |
| **User Inputs & Variables**  User inputs are simply variables that can be used in place of absolute values. They increase the flexibility of your strategy since you can substitute a variable for the period parameter of a simple moving average instead of provide an absolute value.    SMA(9) is how you express a 9 period simple moving average in NinjaScript. If you run a strategy, you would always be using a 9 period simple moving average. At run time, you might want to change this value to 10. User defined inputs accomplish this. If you created an input named "MyInput", you could express the simple moving average as SMA(MyInput). At run time, you can then configure your strategy by setting the value of "MyInput" to whatever value you like. In addition, user inputs are required when [optimizing a strategy](https://ninjatrader.com/es/support/helpGuides/nt8/optimize_a_strategy.htm).    User variables (not to be confused with inputs) behave in the same manner with the exception that they can not be configured when starting a strategy but can only be set programmatically during run time.    •User inputs are created from the [Builder screen](https://ninjatrader.com/es/support/helpGuides/nt8/builder_screens.htm)  •User variables can be set in the strategy logic through the **Condition Builder** (see the sections above)    The following is an example and represents one of many possible combinations, the example demonstrates the use of a user input however the sample approach applies to user variables.    1. Expand the **Price** category and select the **Close**.  2. Expand the **Indicator** category and select the **SMA** indicator  3. Select the **greater** relational operator  4. Set the **Period** parameter to a user defined input by pressing the "**Set**" button (green arrow) to open the **Value** window    Strategy_Builder_CB6  5. Expand the **User input** category and select the value **MAPeriod** and press the **OK** button    Strategy_Builder_CB7    6. The Condition Builder will now look as per the image below with the user input "MAPeriod" assigned to the parameter Period. When you apply this strategy to a chart, you will be able to set the value for the user input directly from the UI which will then be used to drive the SMA indicator.    Strategy_Builder_CB8    Once the **OK** button is pressed, a condition is created that would translate to the following:    **"Current closing price is greater than the user defined Period simple moving average"** |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?condition_builder2.htm#HowToUseUserDefinedInputsAndVariables)

tog_minus        [How to create a cross over condition](javascript:HMToggle('toggle','HowToCreateACrossOverCondition','HowToCreateACrossOverCondition_ICON'))

|  |
| --- |
| **Cross Over Conditions**  You can check for either a **CrossAbove** or **CrossBelow** condition with a user defined look back period. The look back period sets the number of bars to look back to check for the cross over condition.    The following is an example and represents one of many possible combinations.    1. Expand the **Indicator** category and select the **EMA** indicator  2. Set the **Period** parameter to the desired value ("9" is used in this example)  3. Expand the **Indicator** category and select the **EMA** indicator  4. Set the **Period** parameter to the desired value ("20" is used in this example)  5. Select **CrossAbove** relational operator  6. Set the **Look back period**    Strategy_Builder_CB9  Once the **OK** button is pressed, a condition is created that would translate to the following:    **"9 period exponential moving average crosses above the 20 period exponential moving average in the last bar"** |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?condition_builder2.htm#HowToCreateACrossOverCondition)

tog_minus        [How to use indicator inputs in other indicators](javascript:HMToggle('toggle','HowToUseIndicatorInputsInOtherIndicators','HowToUseIndicatorInputsInOtherIndicators_ICON'))

|  |
| --- |
| **Indicator on Indicator**  You can use indicators as input for other indicators ... actually, you can nest indicators within indicators infinitely if you really wanted to!    The following example is an example of applying a simple moving average (**SMA**) to a 14 period **ADX** indicator and is one of many possible combinations.    1. Expand the **Indicator** category and select the **SMA** indicator  2. Set Input series to the **ADX** indicator by pressing the "**Edit Input**" button to open the **Value** window  3. Select the **ADX** indicator and set any properties in the **Parameters** window    Strategy_Builder_CB10  3. Select the **ADX** indicator and set any properties in the **Properties** window  4. Press the **OK**button    Strategy_Builder_CB11  5. Once you have pressed the **OK** button, you will notice on the left lower window, the "Input series" parameters has now been set to the **ADX**(14) which is the 14 period **ADX** indicator.    Strategy_Builder_CB12 |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?condition_builder2.htm#HowToUseIndicatorInputsInOtherIndicators)

tog_minus        [How to check for volume expansion](javascript:HMToggle('toggle','HowToCheckForVolumeExpansion','HowToCheckForVolumeExpansion_ICON'))

|  |
| --- |
| **Checking for Volume Expansion**  You can compare if the current bar's volume is greater than the prior bar's volume plus an offset amount.    The following is an example and represents one of many possible combinations.    1. Expand the **Indicator** category and select the **VOL** indicator  2. Expand the **Indicator** category and select the **VOL** indicator  3. Select the **greater than or equal**relational operator  4. Set the **Bars ago**parameter to a value of "1"  5. Set **Offset type** parameter to **Percent**  6. Set the **Offset** parameter to a value of "3" - *3 equals 300% percent here, i.e. 10% would be 0.1*    Strategy_Builder_CB13  Once the **OK** button is pressed, a condition is created that would translate to the following:    **"Current value of Volume is greater than or equal to the value of Volume of 1 bar ago + 300%"** |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?condition_builder2.htm#HowToCheckForVolumeExpansion)

tog_minus        [How to create market position comparisons](javascript:HMToggle('toggle','HowToCreateMarketPositionComparisons','HowToCreateMarketPositionComparisons_ICON'))

|  |
| --- |
| **Creating Market Position Comparisons**  You can compare strategy state information such as but not limited to current market position or current position size.    The following is an example and represents one of many possible combinations.    1. Expand the **Strategy** category and select **Current market position**.  2. Expand the **Strategy** category and select **Market position**  3.Select the **equals to** relational operator  4. Select **Flat** from the Market position dropdown under Misc    Strategy_Builder_CB14    Once the **OK** button is pressed, a condition is created that would translate to the following:    **"Current market position equals flat"** |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?condition_builder2.htm#HowToCreateMarketPositionComparisons)

tog_minus        [How to create time comparisons](javascript:HMToggle('toggle','HowToCreateTimeComparisons','HowToCreateTimeComparisons_ICON'))

|  |
| --- |
| **Creating Time Comparisons**  You can compare a bar's time data to a user defined time or date value.    The following is an example and represents one of many possible combinations.    Note: Time series represents a collection of bar Date / Time values of a bar series    1. Expand the **Time** category and select **Time series**  2. Expand the **Time** category and select **Time series**  3. Select the **greater than or equal**relational operator  4. Set the **Time** parameter to a user defined value of "10:00"    Strategy_Builder_CB15    Once the **OK** button is pressed, a condition is created that would translate to the following:    **"Current bar's time is greater or equal to 10:00 AM"** |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?condition_builder2.htm#HowToCreateTimeComparisons)

tog_minus        [How to negate a condition](javascript:HMToggle('toggle','HowToNegateACondition','HowToNegateACondition_ICON'))

|  |
| --- |
| **Negating a Condition**  You can also negate a condition, so allowing for example to have a certain filter or technical indicator setup being the opposite and evaluate to false.    The following is an example and represents one of many possible combinations.    1. Expand the **Misc** category and select the **Cross above**  2. Click the **Series 1** input field and select the **DEMA** indicator as series for the cross comparison to use  3. Expand the **Misc** category and select the **False**  4. Select the equals relational operator    Strategy_Builder_CB16  Strategy_Builder_CB17    Once the **OK** button is pressed, a condition is created that would translate to the following:    **"The DEMA(14) indicator has not been crossed by the Close price within the last 10 bars"** |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?condition_builder2.htm#HowToNegateACondition)

|  |  |
| --- | --- |
| **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) > [Attributes](https://ninjatrader.com/es/support/helpGuides/nt8/attributes.htm) >  **DisplayAttribute** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/categoryorderattribute.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/attributes.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascriptpropertyattribute.htm) |

**Definition**

Determines how the following declared property display on the NinjaTrader UI's property grid.

|  |
| --- |
| **Note**: The **DisplayAttribute** object is a general purpose attribute made available from the .NET Framework. The information on this page is written to demonstrate how you may use this object within NinjaScript conventions used with the NinjaTrader UI's property grid (e.g., an indicator dialog).  There are more methods and properties that you can learn about from **MSDN's** [DisplayAttribute Class](https://msdn.microsoft.com/en-us/library/system.componentmodel.dataannotations.displayattribute(v=vs.110).aspx" \t "_blank) which are**NOT** covered in this topic; as such there is **NO** guarantee they will work with the NinjaTrader UI's property grids. |

**Syntax**

[Display(Name=string)]  
[Display(Description=string)]  
[Display(GroupName=string)]  
[Display(Order=int)]

|  |
| --- |
| **Warning**:  The "**Name**" parameter **MUST** be unique for each property of a particular object.  Sharing the same **Name** can have undesirable consequences on various features of the property grid. |

**Parameters**

|  |  |
| --- | --- |
| Name | A string which sets the text used to display the property on the UI |
| Description | A string which sets the tool tip used to describe the property from the UI    **Note**:  Expandable properties will **NOT** display a tool tip (e.g., [SimpleFont](https://ninjatrader.com/es/support/helpGuides/nt8/simplefont_class.htm), [Stroke](https://ninjatrader.com/es/support/helpGuides/nt8/stroke_class.htm), or any custom component which are a type of an ExpandableObjectConverter) |
| GroupName | A string which sets a name that is used to group various properties in the UI. If no GroupName is specified, properties will be listed in the generic "Parameters" section. |
| Order | An int which sets the sequence the property is categorized in relation to other properties in the UI. |

|  |
| --- |
| **Tips**:  1.Multiple named parameters can be written separated by a comma during a single declaration as demonstrated in the example below.  2.You may have noticed the default NinjaTrader types such as indicators or strategies use a "ResourceType = typeof(Custom.Resource)" property in the DisplayAttribute.  This is done for localization purposes, so the default NinjaTrader UI translates to other supported international languages, but is not required for your custom NinjaScript types.   The ResourceType property can be safely ignored and left out in your custom development. |

**Examples**

| ns | |
| --- | --- |
| #region Properties     // set how the property displays from the UI property grid [Display(Name="My Period", Order=1, GroupName="My Parameters")] public int MyPeriod { get; set; }   #endregion | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Attributes](https://ninjatrader.com/es/support/helpGuides/nt8/attributes.htm) >  **NinjaScriptPropertyAttribute** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/displayattribute.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/attributes.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/rangeattribute.htm) |

**Definition**

Determines if the following declared property should be included in the NinjaScript object's constructor as a parameter.  This is useful if you plan on calling a NinjaScript object from another (e.g., calling a custom indicator from a strategy) or customizing the display parameter data on a grid or from a chart.

|  |
| --- |
| **Warning**:  Only types which can be[Xml Serialized](https://ninjatrader.com/es/support/helpGuides/nt8/xmlignoreattribute.htm) should be marked as a **NinjaScriptAttribute**, otherwise you may run into errors when persisting values in various scenarios (e.g., saving workspace, or running [Strategy Optimizations](https://ninjatrader.com/es/support/helpGuides/nt8/optimize_a_strategy.htm)).  Should you have a property you wish to use as user defined input, you will need to implement a secondary simple type (such as an int or string) as the value to be serialized as user input. Please see the example below which demonstrates using a simple type as the **NinjaScriptProperty** against types which cannot be serialized |

**Syntax**

[NinjaScriptProperty]

**Parameters**

This object contains no parameters

**Examples**

| ns **Basic usage of NinjaScriptProperty** |
| --- |
| #region Properties         // set NinjaScriptProperty to ensure this property is used when calling from another object [NinjaScriptProperty] public bool MyBool   { get; set; }   // do not set NinjaScriptProperty since this property is not required to call // nor do we wish to display it on the chart label public int MyInt { get; set; }   #endregion |

| ns **Using a simple type as the NinjaScriptProperty against types which cannot be serialized** | |
| --- | --- |
| [XmlIgnore] // cannot serialize type of TimeSpan, use the BeginTimeSpanSerialize object to persist properties       [Browsable(false)] // prevents this property from showing up on the UI public TimeSpan BeginTimeSpan { get; set; }   // users will configure this "string" as the TimeSpan which will be set as a TimeSpan object used in data processing [NinjaScriptProperty] [Display(Name = "Begin TimeSpan", GroupName = "NinjaScriptStrategyParameters", Order = 1)] public string BeginTimeSpanSerialize {   get { return BeginTimeSpan.ToString(); }   set { BeginTimeSpan = TimeSpan.Parse(value); } } | |
| **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) >  **Attributes** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/totime.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/browsableattribute.htm) |

The following section documents both .NET native and NinjaScript custom [attributes](https://msdn.microsoft.com/en-us/library/5x6cd29c(v=vs.110).aspx" \t "_blank) which are commonly used to define the behavior of a NinjaScript property or object.  The **attributes** outlined in the section are primarily used to customize how properties display on the UI, but may also control or how the object is compiled and executed at run time.

|  |
| --- |
| **Notes**:  1.The .NET Framework supplies many other pre-defined [system attributes](https://msdn.microsoft.com/en-us/library/2e39z096.aspx" \t "_blank) which can technically be used for custom NinjaScript programming, but are **NOT** covered in this section and therefore are considered **unsupported**.  3rd party developers are encourage to explore additional usage, but the resulting behavior **CANNOT** be guaranteed.  2.Not all **attributes** can be applied to all object types.  For example, applying an **attribute** that is defined to target an class will **NOT** compile should you attempt to apply this **attribute** to a type of property. |

**Common Attributes**

|  |  |
| --- | --- |
| [BrowsableAttribute](https://ninjatrader.com/es/support/helpGuides/nt8/browsableattribute.htm) | Determines if a property should be displays in the NinjaTrader UI's property grid |
| [CategoryOrderAttribute](https://ninjatrader.com/es/support/helpGuides/nt8/categoryorderattribute.htm) | Determines the sequence in which a NinjaScript object's [Display.GroupName](https://ninjatrader.com/es/support/helpGuides/nt8/displayattribute.htm) categories are arranged in relation to other categories in the UI. |
| [DisplayAttribute](https://ninjatrader.com/es/support/helpGuides/nt8/displayattribute.htm) | Determines how a property is displays on the NinjaTrader UI's property grid. |
| [NinjaScriptPropertyAttribute](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascriptpropertyattribute.htm) | Determines if a property should be included in the NinjaScript object's constructor as a parameter |
| [RangeAttribute](https://ninjatrader.com/es/support/helpGuides/nt8/rangeattribute.htm) | Determines if the value of a property is valid within a specified range |
| [XmlIgnoreAttribute](https://ninjatrader.com/es/support/helpGuides/nt8/xmlignoreattribute.htm) | Determines if a property participates in the XML serialization routines (saving workspaces or templates) |

**Applying Attributes**

Attributes are applied directly before the property, method, or class, and are identified by wrapping brackets:

| ns |
| --- |
| [AnExampleAttribute] // a pseudo-attribute demonstrating how to target an object public object AnExampleProperty // the property that is being targeted { get; set; } |

|  |  |
| --- | --- |
| **Tip**:   Conventionally, the suffix "attribute" is provided to the object's name to help determine that is an **attribute**, however C# does not require you to specify the full name of an **attribute**.  For example **DisplayAttribute()** will compile the same as **Display(**). | |
| **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) > [Attributes](https://ninjatrader.com/es/support/helpGuides/nt8/attributes.htm) >  **RangeAttribute** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascriptpropertyattribute.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/attributes.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/typeconverterattribute.htm) |

**Definition**

Determines if the value of the following declared property is valid within a specified range.  These values are checked when the NinjaScript object has reached[State.Configure](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm).  For configuration through the UI (e.g., the user has selected Apply or OK to configure the value from the indicator dialog box) and determines to be invalid, the value will be automatically rounded to the nearest minimum or maximum value. Should the property be set as a [NinjaScriptAttribute](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascriptpropertyattribute.htm) and called from a hosting NinjaScript object and determines to be invalid, an exception will be thrown and the hosted indicator will NOT execute.

|  |
| --- |
| **Note**: The **RangeAttribute** object is a general purpose attribute made available from the .NET Framework.  The information on this page is written to demonstrate how you may use this object within NinjaScript conventions used for the NinjaTrader UI's property grid (e.g., an indicator dialog).  There are more methods and properties that you can learn about from **MSDN's** [RangeAttribute Class](https://msdn.microsoft.com/en-us/library/system.componentmodel.dataannotations.rangeattribute(v=vs.110).aspx" \t "_blank) which are**NOT** covered in this topic; as such there is **NO** guarantee they will work with the NinjaTrader UI's property grids. |

**Syntax**

[Range(int minimum, int maximum)]  
[Range(double minimum, double maximum)]  
[Range(type type, string minimum, string aximum)]

**Parameters**

|  |  |
| --- | --- |
| maximum | Defines the highest allowed value the user can set for the property |
| minimum | Defines the lowest allowed value the user can set for the property |
| type | The [type](https://msdn.microsoft.com/en-us/library/system.type(v=vs.110).aspx" \t "_blank) of object to test |

**Examples**

| ns | |
| --- | --- |
| #region Properties   // set range between 1 and the highest possible integer [Range(1, int.MaxValue)] public int Myint { get; set; }         //set range between .001 and 1 [Range(.001, 1.0)] public double MyDouble { get; set; }             // set range as a type DateTime between these dates [Range(typeof(DateTime), "01/01/1990", "12/31/2015")] public DateTime MyTime { get; set; }   #endregion | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) > [Attributes](https://ninjatrader.com/es/support/helpGuides/nt8/attributes.htm) >  **CategoryOrderAttribute** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/browsableattribute.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/attributes.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/displayattribute.htm) |

**Definition**

Determines the sequence in which a NinjaScript object's [Display.GroupName](https://ninjatrader.com/es/support/helpGuides/nt8/displayattribute.htm) categories are arranged in relation to other categories in the UI.   The default behavior will display each GroupName of an object in alphabetical order, however this behavior can be changed by defining the **CategoryOrder** attribute before the object's declaration.

|  |
| --- |
| **Notes**:  •The **CategoryOrder** attribute is **ONLY** valid on class-level declarations.  •Categories with values less than 1,000,000 appear at the very top of the property grid (excluding the Strategy Analyzer "General" category)  •NinjaTrader UI reserves using values ending in 000, 500 and the values documented below are subject to change  •If you wish to inject your category between a standard NinjaScript category, please refer to the table below to locate the appropriate position (e.g., to set a property after "Data Series" and before the "Setup" use value of 2,000,001) |

**NinjaScript Indicators**

The follow table applies for Indicators configured from a Chart Indicator, Market Analyzer Indicator Column, or SuperDOM Indicator:

|  |  |
| --- | --- |
| Parameters | 1000000 |
| Data Series | 2000000 |
| Time Frame | 3000000 |
| Setup | 4000000 |
| Visual | 5000000 |
| Lines | 6000000 |
| Plots | 7000000 |

**NinjaScript Strategies**

The following table applies to Chart Strategies, Control Center Strategies Grid, and the Strategy Analyzer

|  |  |
| --- | --- |
| Parameters | 1000000 |
| Data Series | 2000000 |
| Time Frame | 3000000 |
| Setup | 4000000 |
| Historical Fill Processing | 5000000 |
| Optimize | 6000000 |
| Order Handling | 7000000 |
| Order Properties | 8000000 |

|  |
| --- |
| **Note**:  The Strategy Analyzer "General" category is purposely excluded from this table and will always show on the top of the parameter grid. |

**Syntax**

[Gui.CategoryOrder(string category, int order)]

|  |
| --- |
| **Warning**:  Attempting to modify the default NinjaScript Category ordering is **NOT** supported.  Trying to do so may result in unexpected outcomes. |

**Parameters**

|  |  |
| --- | --- |
| category | A string identifying the [GroupName](https://ninjatrader.com/es/support/helpGuides/nt8/displayattribute.htm) to be categorize |
| order | An int determining the sequence the Category displays |

**Examples**

| ns | |
| --- | --- |
| [Gui.CategoryOrder("My Strings", 1)] // display "My Strings" first [Gui.CategoryOrder("My Bools", 2)] // then "My Bools" [Gui.CategoryOrder("My Ints", 3)] // and finally "My Ints" public class MyCustomIndicator : Indicator {   #region Properties       [Display(GroupName="My Ints")]   public int MyCustomInt   { get; set; }     [Display(GroupName="My Bools")]   public bool MyCustomBool   { get; set; }     [Display(GroupName="My Strings")]   public string MyCustomString   { get; set; }     #endregion } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) >  **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]; } |