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

**Definition**

Informs the Strategy Analyzer how many iterations of optimizing it needs to do.

**Property Value**

An int value.

**Syntax**

NumberOfIterations

**Examples**

| ns |
| --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)         Name = "MyOptimizer";     else if (State == State.Configure && Strategies.Count > 0)         NumberOfIterations = 1; }   |  |  | | --- | --- | | **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Optimizer](https://ninjatrader.com/es/support/helpGuides/nt8/optimizer.htm) >  **OnOptimize()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/numberofiterations.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/optimizer.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/optimizationparameters.htm) |   **Definition**  This method must be overridden in order to optimize a strategy. This method is called once per optimization run (not once per iteration).    **Method Return Value**  This method does not return a value.    **Syntax** You must override the method in your Optimizer with the following syntax.    **protected override void OnOptimize()** **{**  **}**        **Examples**   | ns | | | --- | --- | | protected override void OnOptimize()  {      // If there is no optimization objective, return      if (Strategies[0].OptimizationParameters.Count == 0)          return;        // Optimizer logic  } | | | **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) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Optimizer](https://ninjatrader.com/es/support/helpGuides/nt8/optimizer.htm) >  **RunIteration()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/optimizationparameters.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/optimizer.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/supportsmultiobjectiveoptimiza.htm) |   **Definition**  Runs an iteration of backtesting for the optimizer    **Method Return Value**  This method does not return a value.    **Syntax**  RunIteration()        **Examples**   | ns | | | --- | --- | | protected override void OnOptimize()  {      // Optimizer logic      RunIteration();  } | | | **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) >  **SupportsMultiObjectiveOptimization** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/runiteration.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/optimizer.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/performance_metrics.htm) |   **Definition**  Informs the Strategy Analyzer if this Optimizer can do multi-objective optimizations.    **Property Value**  A bool value.    **Syntax**  SupportsMultiObjectiveOptimization      **Examples**   | ns | | | --- | --- | | protected override void OnStateChange()  {      if (State == State.SetDefaults)      {           Name = "MyOptimizer";           SupportsMultiObjectiveOptimization = true;      }  } | | | **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) >  **Optimizer** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/optimization_fitness_value.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/numberofiterations.htm) |   Custom Optimizers can be used to optimize your Strategy through different algorithms. These may allow you to make trade offs like being able to find adequate results quickly as opposed to trying to find the absolute best result but through a time consuming process. The methods and properties covered in this section are unique to custom Optimizer development.    **In this section**   |  |  | | --- | --- | | [NumberOfIterations](https://ninjatrader.com/es/support/helpGuides/nt8/numberofiterations.htm) | Informs the Strategy Analyzer how many iterations of optimizing it needs to do. | | [OnOptimize()](https://ninjatrader.com/es/support/helpGuides/nt8/onoptimize.htm) | This method must be overridden in order to optimize a strategy. | | [OptimizationParameters](https://ninjatrader.com/es/support/helpGuides/nt8/optimizationparameters.htm) | The optimization parameters selected for the optimization run. | | [RunIteration()](https://ninjatrader.com/es/support/helpGuides/nt8/runiteration.htm) | Runs an iteration of backtesting for the optimizer. | | [SupportsMultiObjectiveOptimization](https://ninjatrader.com/es/support/helpGuides/nt8/supportsmultiobjectiveoptimiza.htm) | Informs the Strategy Analyzer if this Optimizer can do multi-objective optimizations. | | **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) >  **IsInstantiatedOnEachOptimizationIteration** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/isfilllimitontouch.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/istradinghoursbreaklinevisible.htm) |   **Definition**  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).    The **default behavior** is to re-instantiate the strategy for each optimization backtest run. However, the process of re-instantiating a strategy requires more time and computer resources to return results, which could impact the amount of time it takes to run an optimization.  When **false**, the strategy is re-used to save time and computer resources.  Under this design, internal properties are reset to default values after each iteration, but it is possible that user-defined properties and other custom resources may carry their state over from the previous iteration into a new backtest run.  To take advantage of performance optimizations, developers may need to reset class level variables in the strategy otherwise unexpected results can occur.     |  | | --- | | **Note**:  If you choose to take advantage of the performance benefits during strategy optimization by setting the **IsInstantiatedOnEachOptimizationIteration** property to **false**, any objects you create in your code **MUST** be reset duringthe appropriate**State**within the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm)method.  Please see the example below on "*Manually resetting class level variables to take advantage of Strategy Analyzer optimizer performance benefits*". |       **Property Value**  This property returns **true** if the strategy is not recycled; otherwise, **false**. Default set to **true**.     |  | | --- | | **Warning**:  This property should **ONLY** bet set from the [OnStateChange()](https://ninjatrader.com/es/support/helpGuides/nt8/onstatechange.htm) method during **State.SetDefaults** or **State.Configure** |       **Syntax**  IsInstantiatedOnEachOptimizationIteration     |  | | --- | | **Tip**:  The default NinjaTrader indicators and strategies have been optimized to take advantage of performance optimizations as their resources are setup >= **State.Configure**.  Please see the default system indicators and strategies for an idea of how you may improve your strategy and indicator performance, or you may also reference the example code below. |       **Examples**   | ns **Using IsInstantiatedOnEachOptimizationIteration to reset class level variables** | | --- | | // A custom trades dictionary is created when strategy is instantiated // since we later set "IsInstantiatedOnEachOptimizationIteration" to true, // we are guaranteed to start with a new object on each optimization run private Dictionary<DateTime, string> myTrades = new Dictionary<DateTime, string>();   protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name       = "My Optimization Test 1";     Description = "Demonstrates using IsInstantiatedOnEachOptimizationIteration to reset a class level variable";     Fast       = 10;     Slow       = 25;       // setting to true so our custom trades dictionary is reset on each optimization run (comes with a performance penalty)     // This is the default behavior.     IsInstantiatedOnEachOptimizationIteration = true;   }     else if (State == State.Terminated)   {     // Print the number of trades at the end of the optimization     if (myTrades != null)     {         // if we set "IsInstantiatedOnEachOptimizationIteration" to false (so not using the default of true), the values here would be unexpected         // since the custom trade dictionary was never explicitly reset at the end of each optimization         Print(myTrades.Count);     }   } }   protected override void OnBarUpdate() {   if (CurrentBar < BarsRequiredToTrade)     return;     if (CrossAbove(SMA(Fast), SMA(Slow), 1))   {     EnterLong();     myTrades.Add(Time[0], "long");     }   else if (CrossBelow(SMA(Fast), SMA(Slow), 1))   {     EnterShort();     myTrades.Add(Time[0], "short");   } }   [Range(1, int.MaxValue), NinjaScriptProperty] [Display(Name = "Fast", GroupName = "NinjaScriptStrategyParameters", Order = 0)] public int Fast { get; set; }   [Range(1, int.MaxValue), NinjaScriptProperty] [Display(Name = "Slow", GroupName = "NinjaScriptStrategyParameters", Order = 1)] public int Slow { get; set; } |        | ns **Manually resetting class level variables to take advantage of Strategy Analyzer optimizer performance benefits** | | | --- | --- | | // A custom trades dictionary is declared when strategy is first optimized, // but not instantiated until later in State.DataLoaded, private Dictionary<DateTime, string> myTrades;   // examples of other fields which need to be reset private double myDouble; private bool myBool; private DateTime myDateTime; private Order myOrderObject; private Brush myBrushObject; private SMA mySMAIndicator; private Array myIntArray; private List<object> myList; private Series<double> mySeries;   protected override void OnStateChange() {   if (State == State.SetDefaults)   {     Name = "My Optimization Test 2";     Description = "Demonstrates manually resetting a class level variable without re-instantiating the strategy";     Fast = 10;     Slow = 25;           // in this case, we do not need to re-instantiate the strategy after each optimization     // because we are explicitly resetting the custom trade dictionary in State.DataLoaded     // This design of re-using the strategy instance comes with performance benefits     IsInstantiatedOnEachOptimizationIteration = false;   }     else if (State == State.DataLoaded)   {     // re-create custom trade dictionary on each optimization run     // we are guaranteed to start with a new object on each optimization run     if (myTrades != null)       myTrades.Clear();     else       myTrades = new Dictionary<DateTime, string>();           //Any strategy defaults which are maintained do not need to be reset if they are not mutable as the strategy runs.     //Any strategy state that would be mutable after State.SetDefaults needed to be reset for the next run.     myDouble = double.MinValue;     myBool = false;     myDateTime = DateTime.MinValue;     myOrderObject = null;     myBrushObject = null;     mySMAIndicator = SMA(14);           if (myIntArray != null)         Array.Clear(myIntArray, 0, myIntArray.Length);     else         myIntArray = new int[20];           if (myList != null)         myList.Clear();     else         myList = new List<object>();           mySeries = new Series<double>(this);   } }   protected override void OnBarUpdate() {   if (CurrentBar < BarsRequiredToTrade)     return;     if (CrossAbove(SMA(Fast), SMA(Slow), 1))   {     EnterLong();     myTrades.Add(Time[0], "long");     }   else if (CrossBelow(SMA(Fast), SMA(Slow), 1))   {     EnterShort();     myTrades.Add(Time[0], "short");   } }   [Range(1, int.MaxValue), NinjaScriptProperty] [Display(Name = "Fast", GroupName = "NinjaScriptStrategyParameters", Order = 0)] public int Fast { get; set; }   [Range(1, int.MaxValue), NinjaScriptProperty] [Display(Name = "Slow", GroupName = "NinjaScriptStrategyParameters", Order = 1)] public int Slow { get; set; } | | | **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Common](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) >  **OnStateChange()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/marketdeptheventargs.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/common.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/setstate.htm) |   **Definition**  An event driven method which is called whenever the script enters a new [State](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm). The **OnStateChange()** method can be used to configure script properties, create one-time behavior when going from historical to real-time, as well as manage clean up resources on termination.       |  | | --- | | **Notes**:  •Viewing any UI element which lists NinjaScript classes (such as the Indicators or Strategies window, a chart's Chart Style dropdown menu, etc.) will initialize all classes of that Type when it is opened, which causes each script to enter **State.SetDefaults**, even if it is not actively configured or running in any window. It is important to keep this in mind when adding logic within **State.SetDefaults** in **OnStateChange()**, as this logic will be processed each time the script is initialized. For example, opening the Indicators window will trigger **State.SetDefaults** for all indicators in the system, and closing the Indicators window will trigger**State.Terminated** for all Indicators. In addition, disconnecting or connecting to a data provider can cause State transitions for any currently active scripts. Further discussion of this aspect of the state change model can be found via *[Understanding the lifecycle of your NinjaScript objects](https://ninjatrader.com/es/support/helpGuides/nt8/understanding_the_lifecycle_of.htm)*.  •When an indicator is configured on a chart while a Compile is taking place in the NinjaScript Editor, it can appear that the script passes through **State.Terminated**. However, this is the result of a copy of the script being initialized at compile-time, NOT the result of the indicator on the chart being disabled and re-initialized. |       **Related Methods and Properties**   |  |  | | --- | --- | | [SetState()](https://ninjatrader.com/es/support/helpGuides/nt8/setstate.htm) | Method is used for changing the State of any running NinjaScript object. | | [State](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm) | Represents the current progression of the object as it advances from setup, processing data, to termination. |       **Method Return Value**  This method does not return a value.      **Syntax** See example below. The NinjaScript wizards automatically generate the method syntax for you.    Possible states are:   |  |  |  | | --- | --- | --- | | **State Name** | **This state is called when** | **This state is where you should** | | State.SetDefaults | **SetDefaults** is always called when displaying objects in a UI list such as the Indicators dialogue window since temporary objects are created for the purpose of UI display | •Keep as lean as possible  •Set default values (pushed to UI) | | State.Configure | **Configure** is called after a user adds an object to the applied list of objects and presses the OK or Apply button.  This state is called only once for the life of the object. | •Add additional data series via [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm)  •Declare custom resources | | State.Active | **Active** is called once after the object is configured and is ready to process data (DrawingTools could see multiple calls as internally an object for hit testing is cloned) | •Used for objects such as [Share Service](https://ninjatrader.com/es/support/helpGuides/nt8/share_service.htm) which do not process price series data  •Indicate the object is ready to being processing information | | State.DataLoaded | **DataLoaded** is called only once after all data series have been loaded. | •Use for logic that needs to access data related objects like Bars, Instruments, BarsPeriod, TradingHours or instantiating indicators  •Notify that all data series have been loaded  •Initialize any class level variables (including custom [Series<T>](https://ninjatrader.com/es/support/helpGuides/nt8/seriest.htm) objects) | | State.Historical | **Historical**is called once the object begins to process historical data. This state is called once when running an object in real-time. This object is called multiple times when running a backtest optimization and the property [IsInstantiatedOnEachOptimizationIteration](https://ninjatrader.com/es/support/helpGuides/nt8/isinstantiatedoneachoptimizationiteration.htm) is true (default behavior) | •Notify that the object is processing historical data | | State.Transition | **Transition** is called once as the object has finished processing historical data but before it starts to process realtime data. | •Notify that the indicator or strategy is is transitioning to realtime data  •Prepare realtime related resources | | State.Realtime | **Realtime** is called once when the object begins to process realtime data. | •Notify that the indicator or strategy is processing realtime data  •Execute realtime related logic | | State.Terminated | **Terminated** is called once when the object terminates. | •Notify the object is shutting down  •Use to clean up/dispose of resources |       **Active States vs Data Processing States**  After **State.Configure,**each type of NinjaScript type has its own state management system which can be classified under two categories:    •**Active state:**  State.Active  •**Data Processing states:**State.DataLoaded, State.Historical, State.Transition, State.Realtime    The table below lists each NinjaScript type and it's designed state management system:     |  |  | | --- | --- | | **NinjaScript Type** | **State Management System** | | AddOns\* | Active state | | BarTypes | Active state | | ChartStyles | Active state | | DrawingTools | Active state | | Indicators | Data Processing states | | ImportTypes | Active state | | Market Analyzer Columns | Data Processing states | | OptimizationFitnesses | Active state | | Optimizers | Active state | | PerformanceMetrics | Active state | | ShareServices | Active state | | Strategies | Data Processing states | | SuperDOM Columns | Active state |        |  | | --- | | **Tips:**  •Resources created in **State.Configure** and not disposed of immediately will be kept and utilized if the NinjaScript object resides in grids (e.g. Strategy tab on Control Center), even if it is not enabled. Try to create resources in**State.Historical** or **State.DataLoaded** instead, if possible.  •**State.Historical** is called multiple times when running a backtest [optimization](https://ninjatrader.com/es/support/helpGuides/nt8/optimize_a_strategy.htm) on a strategy and the property "[IsInstantiatedOnEachOptimizationIteration](https://ninjatrader.com/es/support/helpGuides/nt8/isinstantiatedoneachoptimizationiteration.htm)" is **true** (default behavior).  • Scripts that require [Calculate](https://ninjatrader.com/es/support/helpGuides/nt8/calculate.htm) to be set by the developer must set this property in **State.Historical** in order to ensure that if this script is a child (hosted) that the parent.Calculate property which is adopted by the child is overridden again.  •When instantiating indicators in a [Multi-Series script](https://ninjatrader.com/es/support/helpGuides/nt8/multi-time_frame__instruments.htm) in OnStateChange, the input any hosted indicator is running on should be explicitly stated (since a specific [BarsInProgress](https://ninjatrader.com/es/support/helpGuides/nt8/barsinprogress.htm) is not guaranteed) |       **Examples**   | ns | | | --- | --- | | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     // Calculate once at the end of every single bar     Calculate = Calculate.OnBarClose;         // Add two plots     AddPlot(Brushes.Blue, "Upper"));     AddPlot(Brushes.Orange, "Lower"));   }     else if (State == State.Configure)   {     // Adds a 5-minute Bars object to the strategy and is automatically assigned     // a Bars object index of 1 since the primary data the strategy is run against     // set by the UI takes the index of 0.             AddDataSeries("AAPL", BarsPeriodType.Minute, 5);       } } | | | **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Optimizer](https://ninjatrader.com/es/support/helpGuides/nt8/optimizer.htm) >  **OnOptimize()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/numberofiterations.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/optimizer.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/optimizationparameters.htm) |   **Definition**  This method must be overridden in order to optimize a strategy. This method is called once per optimization run (not once per iteration).    **Method Return Value**  This method does not return a value.    **Syntax** You must override the method in your Optimizer with the following syntax.    **protected override void OnOptimize()** **{**  **}**        **Examples**   | ns | | | --- | --- | | protected override void OnOptimize()  {      // If there is no optimization objective, return      if (Strategies[0].OptimizationParameters.Count == 0)          return;        // Optimizer logic  } | | | **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); } } | |