|  |  |
| --- | --- |
| **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](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:**  [Operaciones](https://ninjatrader.com/es/support/helpGuides/nt8/operations.htm) > [Analizador de estrategia](https://ninjatrader.com/es/support/helpGuides/nt8/strategy_analyzer.htm) >  **Optimización Multi-Objetivo** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/walk_forward_optimize_a_strate.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy_analyzer.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/ai-generate.htm) |

La optimización multi-objetivo lleva la optimización estándar un paso más allá al permitirle elegir múltiples objetivos para probar. Cuando se devuelven los resultados en lugar de una lista singular de los mejores resultados clasificados de mejor a menor, se le presentará un gráfico. Con objetivos múltiples, no hay un mejor resultado único, sino que depende del operador elegir cuál es la mejor compensación entre dos objetivos. Para ejecutar una optimización de objetivos múltiples, necesitará:

•Acceso a [datos históricos.](https://ninjatrader.com/es/support/helpGuides/nt8/data_by_provider.htm)

•Costumbre NinjaScript \* [estrategia](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm)

•Una comprensión profunda de las capacidades de backtesting y optimización de Strategy Analyzer

|  |
| --- |
| **Consejo** :  Hay varias estrategias de muestra predefinidas que se instalan con NinjaTrader que puede explorar. |

|  |
| --- |
| **Nota**: La propiedad [IncludeTradeHistoryInBacktest](https://ninjatrader.com/es/support/helpGuides/nt8/includetradehistoryinbacktest.htm) se establece en **falso** de forma predeterminada cuando se aplica una estrategia en el **Analizador de estrategias** para la optimización. Esto proporciona un uso de memoria más ágil, pero a expensas de no poder acceder a objetos **comerciales** para intercambios históricos. Por lo tanto, los campos como [SystemPerformance.AllTrades.Count](https://ninjatrader.com/es/support/helpGuides/nt8/alltrades.htm) que dependen de referencias a objetos **Trade** no tendrán tales referencias para trabajar. Si desea guardar estos objetos como referencia en su código, puede establecer **IncludeTradeHistoryInBacktest** en **verdadero** en el estado **Configurar**. Para obtener más información, consulte la página [Trabajar con datos comerciales históricos](https://ninjatrader.com/es/support/helpGuides/nt8/strategyanalyzer_properties_2.htm). |

tog_minus        [Cómo ejecutar una optimización de objetivos múltiples](javascript:HMToggle('toggle','HowtorunaMulti-ObjectiveOptimization','HowtorunaMulti-ObjectiveOptimization_ICON'))

|  |  |  |
| --- | --- | --- |
| **Comience una optimización de objetivos múltiples**  Para ejecutar una **Optimización de objetivos múltiples,** seleccione el **tipo** de Backtest de " **Optimización de objetivos múltiples"** en el panel de configuración del **Analizador de estrategias** .    StrategyAnalyzer_Optimization_MOORun     |  | | --- | | **Nota**: Al hacer la selección, se harán visibles parámetros adicionales para configurar su optimización. |       **Establecer el rango de prueba**  Puede establecer el rango de prueba de los parámetros de estrategia que se probarán haciendo clic izquierdo en el triángulo para expandir los subparámetros de las estrategias.     |  | | --- | | **Nota**: Si no ve el triángulo, asegúrese de que el **tipo de Backtest** esté configurado en " **Optimización de objetivos múltiples** ". |       StrategyAnalyzer_Optimization_Paramters    **Min** . - El valor inicial que desea probar **Max** . - El último valor para probar **Incremento** - El valor de incremento (valor de paso) utilizado para incrementar el valor inicial en cada pasada de optimización posterior    En la imagen de arriba, la entrada "Rápido" tiene un valor inicial (inicial) de 10 y un valor final de 30 con un incremento de 1. Esto significa que el primer valor probado será 10, luego 11, luego 12 hasta 30. La entrada "Lento" tiene un valor inicial de 6, un valor final de 16 con un incremento de 1. En función de esta configuración, se procesarán un total de 200 (20 valores únicos para "Rápido" multiplicado por 10 valores únicos para "Lento") iteraciones de backtest para encontrar la combinación óptima de valores de entrada basada en la mejor aptitud de optimización. |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?multi-objective_optimization.htm#HowtorunaMulti-ObjectiveOptimization)

tog_minus        [Comprender las propiedades de varios objetivos](javascript:HMToggle('toggle','UnderstandingMulti-Objectiveproperties','UnderstandingMulti-Objectiveproperties_ICON'))

|  |  |  |
| --- | --- | --- |
| **Configuración de la aptitud de optimización múltiple**  Además de la propiedad "**Optimizar en**" que se describe a continuación, las propiedades son idénticas a las que se encuentran en la ventana Propiedades de optimización. Consulte la sección " *Descripción de las propiedades de optimización* " de la página [Optimizar una estrategia](https://ninjatrader.com/es/support/helpGuides/nt8/optimize_a_strategy.htm) de la Guía de ayuda para obtener más información.    La optimización de objetivos múltiples se basa en la mejor aptitud de optimización que seleccione. Si establece la propiedad " **Optimizar en** " en "Beneficio neto máximo", "Factor de beneficio máximo" y "Reducción mínima", el optimizador buscará los valores de entrada óptimos basados en esos tres objetivos de optimización física.  Hay más de 10 criterios de optimización diferentes que puede seleccionar y personalizar a través de NinjaScript.    StrategyAnalyzer_Optimization_MultiObjectiveProperties     |  |  | | --- | --- | | Optimizar en ... | Establece la aptitud de optimización para basar los resultados de optimización, al hacer clic con el botón izquierdo en el campo se abrirá la ventana "Editar aptitud de optimización" donde puede habilitar qué aptitud de optimización desea que se pruebe y que esté disponible para el análisis de objetivos múltiples. |       StrategyAnalyzer_Optimization_OptimizationFitnessDialog |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?multi-objective_optimization.htm#UnderstandingMulti-Objectiveproperties)

tog_minus        [Comprender los resultados de múltiples objetivos](javascript:HMToggle('toggle','UnderstandingMultiobjectiveResults','UnderstandingMultiobjectiveResults_ICON'))

|  |
| --- |
| **Comprender los resultados de múltiples objetivos**  **Los** resultados de objetivos múltiples se muestran en un gráfico en lugar de una cuadrícula. La razón por la que usamos un gráfico es con un problema de **objetivos múltiples**, no hay una mejor solución y, en su lugar, debe comparar el equilibrio individual entre dos objetivos a menudo en competencia. Vea la imagen de abajo a la izquierda con algunos datos de muestra, cada optimización se ha realizado y los resultados de cada prueba se trazan en el gráfico. Podemos reducir aún más nuestra solución al mostrar solo los resultados que tienen la mejor compensación entre ambos objetivos conocidos como resultado óptimo de Pareto. En el gráfico de la derecha, la línea dibujada conecta los 5 resultados únicos que son óptimos de Pareto formando la frontera de Paretor. Cualquier resultado que quede atrás de la frontera de Pareto se descarta, dejándonos con las 5 mejores soluciones de compensación entre los dos objetivos.  pareto_frontier_graph  **Usando el gráfico multi-objetivo**  Hay dos selecciones de cuadro combinado para elegir la aptitud de optimización que se representará gráficamente. Podrá elegir cualquier aptitud de optimización que haya habilitado en el campo de optimización en las estrategias de optimización. Consulte la sección de propiedades de optimización multiobjetivo anterior para obtener más información.    StrategyAnalyzer_Optimization_Multi-Objective    Al hacer clic con el botón izquierdo en uno de los puntos, se seleccionará esa ejecución de optimización y NinjaTrader ejecutará una prueba inversa con estos parámetros de estrategia para recuperar los datos comerciales detallados para un análisis posterior. |

[permalink](https://ninjatrader.com/es/support/helpGuides/nt8/index.html?multi-objective_optimization.htm#UnderstandingMultiobjectiveResults)

|  |  |
| --- | --- |
| **Navigation:**  [Operaciones](https://ninjatrader.com/es/support/helpGuides/nt8/operations.htm) > [Ventana de datos históricos](https://ninjatrader.com/es/support/helpGuides/nt8/historical_data_manager.htm) >  **Datos por proveedor** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/loading_historical_data.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/historical_data_manager.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/importing.htm) |

**Comprender los datos proporcionados por su proveedor de conectividad**

NinjaTrader, LLC no es un proveedor de datos de mercado. Nuestros proveedores de conectividad proporcionan datos históricos que ofrecen datos históricos como parte de su servicio. La siguiente tabla muestra todos los proveedores de conectividad compatibles con NinjaTrader, así como los datos históricos y en tiempo real proporcionados por cada uno:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Proveedor de conectividad** | **Datos en tiempo real** | **Datos históricos de marca** | **Datos históricos de oferta / demanda de minutos** | **Datos históricos de oferta / demanda diaria** | **Datos históricos de ticks de oferta / demanda** | **Datos históricos de minutos** | **Datos diarios históricos** | **Marca de tiempo en tiempo real** | **Instrumentos soportados** | **Noticias en tiempo real** | **Tick Replay** | **Datos de marca estampada de oferta / demanda** | **Horario diario de negociación de barras** | **Liquidación ajustada Precio de cierre para barras diarias** |
| **Kinetick**  [**www.kinetick.com**](http://www.kinetick.com/) | SÍ (solo suscripción) | SÍ (solo suscripción) | NO | NO | SÍ | SÍ (solo suscripción) | SÍ | Nativo | E, F, FX, I | SÍ | SÍ | SÍ | Mapa de símbolos específico | SÍ |
| **BarChart** | SÍ | SÍ | NO | NO | NO | SÍ | SÍ | Nativo | E, F, FX, I | NO | SÍ | NO | Mapa de símbolos específico | SÍ |
| **Coinbase** | SÍ | NO | NO | NO | NO | SÍ | SÍ | Nativo | CC | NO | NO | NO | UTC | N/A |
| **Continuo** | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | Nativo | F, I | NO | SÍ | SÍ | Horario comercial extendido | SÍ |
| **Continuum WebAPI** | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | Nativo | F, I, O | NO | SÍ | SÍ | Horario comercial extendido | SÍ |
| **CQG** | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | Nativo | F, I | NO | SÍ | SÍ | Horario comercial extendido | SÍ |
| **eSignal** | SÍ | SÍ | NO | NO | SÍ | SÍ | SÍ | Nativo | E, F, FX, I | NO | SÍ | NO | Mapa de símbolos específico | SÍ |
| **FOREX.com** | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | Nativo | FX | NO | SÍ | N/A | Forex | N/A |
| **FOREX.com/City Index con G2** | SÍ | SÍ | NO | NO | NO | SÍ | SÍ | Nativo | FX | NO | SÍ | N/A | Forex | N/A |
| **FXCM** | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | Nativo | FX, C | NO | SÍ | N/A | Forex | N/A |
| **Interactive Brokers** | SÍ | NO | SÍ | SÍ | NO | SÍ  (solo cuenta en vivo) | SÍ  (solo cuenta en vivo) | Local | C, E, F, FX, I | NO | NO | NO | Horario comercial extendido | NO |
| **IQFeed** | SÍ | SÍ | NO | NO | SÍ | SÍ | SÍ | Nativo | E, F, FX, I | SÍ | SÍ | SÍ | Mapa de símbolos específico | SÍ |
| **Oanda** | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | Nativo | FX | NO | SÍ | N/A | Forex | N/A |
| **Rítmico** | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | SÍ | Nativo | F | NO | SÍ | SÍ | Horario comercial extendido | SÍ |
| **TD Ameritrade** | SÍ | NO | NO | NO | NO | SÍ | SÍ | Local | E, I | SÍ | NO | NO | Horario comercial regular | NO |

C = CFD  
CC = CryptoCurrency

E = Acciones

F = Futuros

FX = Forex  
I = Índices

O = Opciones

**Convertir datos en tiempo real en datos históricos**

NinjaTrader por defecto siempre cargará datos históricos de su proveedor (Recomendado). Sin embargo, si habilita la opción 'Grabar datos en vivo como históricos' en el Centro de control > Herramientas > Opciones > Categoría de datos de mercado, NinjaTrader almacenará los datos entrantes en tiempo real en su PC local si tiene un Gráfico o un **Analizador de mercado** (debe tener una ventana indicadora agregada) ventana abierta. Estos datos se pueden usar como datos históricos. Por ejemplo, si abre un gráfico y deja que se ejecute todo el día, los datos recopilados hoy estarán disponibles como datos históricos cuando abra el mismo gráfico mañana.

|  |
| --- |
| **Advertencia** : La grabación de datos en vivo utiliza más recursos de PC y está destinada a conexiones que **NO** proporcionan datos históricos.  No se recomienda habilitar esta opción mientras se usa un proveedor de datos históricos, ya que puede generar lagunas en los datos. |

**Conexión simultánea con su corredor y un proveedor de datos de mercado**

Si la tecnología de su corredor no admite datos históricos, puede conectarse a un servicio como Kinetick al mismo tiempo que se conecta a su corredor para que pueda recibir datos históricos. Consulte el tema sobre [Conexiones múltiples](https://ninjatrader.com/es/support/helpGuides/nt8/multiple_connections.htm) para obtener información adicional.

|  |  |
| --- | --- |
| **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) >  **StrategyBaseConverter Class** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/stoptargethandling.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/systemperformance.htm) |

**Definition**

A custom [TypeConverter](https://msdn.microsoft.com/en-us/library/system.componentmodel.typeconverter%28v=vs.110%29.aspx" \t "_blank) class handling the designed behavior of a strategy's property descriptor collection.  Use this as a base class for any custom **TypeConverter** you are applying to a strategy class.

|  |
| --- |
| **Notes:**  •A working NinjaScript demo can be found through the reference sample on "[Using a TypeConverter to Customize Property Grid Behavior](http://ninjatrader.com/support/forum/showthread.php?t=97919" \t "_blank)"  •When applying the custom converter, you must fully qualify the name (e.g., "NinjaTrader.NinjaScript.Strategies.MyCustomConveter")  •Additional **TypeConverter** information can be found from the [MSDN documentation](https://msdn.microsoft.com/en-us/library/system.componentmodel.typeconverter%28v=vs.110%29.aspx)  •See also [TypeConverterAttribute](https://ninjatrader.com/es/support/helpGuides/nt8/typeconverterattribute.htm)  •For Indicators, see the [IndicatorBaseConverter](https://ninjatrader.com/es/support/helpGuides/nt8/indicatorbaseconverter.htm) class |

**Relevant base methods**

|  |  |
| --- | --- |
| [TypeConverter.GetProperties()](https://msdn.microsoft.com/en-us/library/system.componentmodel.typeconverter.getproperties(v=vs.110).aspx) | When overriding **GetProperties()**, calling base.GetProperties() ensures that all default property grid behavior works as designed |
| [TypeConverter.GetPropertiesSupported()](https://msdn.microsoft.com/en-us/library/system.componentmodel.typeconverter.getpropertiessupported(v=vs.110).aspx) | In your custom converter class, you must override **GetPropertiesSupported()**and return a value of **true** in order for your custom type converter to work |

**Syntax**

public class StrategyBaseConverter : TypeConverter

|  |
| --- |
| **Warning**:  Failure to apply a type of **StrategyBaseConverter** on an strategy class can result in unpredictable behavior of the standard NinjaTrader WPF property grid. |

|  |
| --- |
| **Tip**: Common strategy functions like Print() are not available to a type converter instance.  To debug a type converter class, you can use the AddOn [Debug Concepts](https://ninjatrader.com/es/support/helpGuides/nt8/alert_and_debug_concepts.htm) or [attach to a debugger](https://ninjatrader.com/es/support/helpGuides/nt8/visual_studio_debugging.htm) (recommended) |

**Examples**

| ns | |
| --- | --- |
| //This namespace holds Strategies in this folder and is required. Do not change it. namespace NinjaTrader.NinjaScript.Strategies {   // When applying the type converter, you must fully qualify the name   [TypeConverter("NinjaTrader.NinjaScript.Strategies.MyCustomConveter")]   public class MyCustomStrategy : Strategy   {     protected override void OnStateChange()     {         if (State == State.SetDefaults)         {           Name                             = "MyCustomStrategy";         }     }       protected override void OnBarUpdate()     {         //Add your custom strategy logic here.     }   }     // custom converter class for strategies   public class MyCustomConveter : StrategyBaseConverter   {     // A general TypeConveter method used for converting types     public override PropertyDescriptorCollection GetProperties(ITypeDescriptorContext context, object component, Attribute[] attrs)     {         // sometimes you may need the strategy instance which actually exists on the grid         MyCustomStrategy strategy = component as MyCustomStrategy;           // base.GetProperties ensures we have all the properties (and associated property grid editors)         // NinjaTrader internal logic handles for a given strategy         PropertyDescriptorCollection propertyDescriptorCollection = base.GetPropertiesSupported(context)                 ? base.GetProperties(context, component, attrs) : TypeDescriptor.GetProperties(component, attrs);           if (strategy == null || propertyDescriptorCollection == null)           return propertyDescriptorCollection;           // example of why you may need the instance that exists on the grid....         if (strategy.EntryHandling == EntryHandling.UniqueEntries)         {           // do something in the event a property contains some value...         }           // Loop all of the properties of the strategy         foreach (PropertyDescriptor property in propertyDescriptorCollection)         {           // do something with a specific property             // cannot call Print() here           // but you can call the static Output window "Process()"           NinjaTrader.Code.Output.Process(property.Name, PrintTo.OutputTab1);         }           // must return the collection after making changes         return propertyDescriptorCollection;     }       // Important:  This must return true otherwise the type converter will not be called     public override bool GetPropertiesSupported(ITypeDescriptorContext context)     { return true; }   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) >  **SystemPerformance** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/strategybaseconverter.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/alltrades.htm) |

**Definition**

The SystemPerformance object holds all trades and trade performance data generated by a strategy.

|  |
| --- |
| **Notes**:  •A NinjaScript strategy can generate both synthetic trades (historical backtest trades) and real-time trades executed on a real-time data stream. If you wish to access only real-time trades, access the "RealTimeTrades" collection  •The first trade of the "RealTimeTrades" collection will contain a synthetic entry execution if the strategy was **NOT** flat at the time you start the strategy.  •These properties require that [IncludeTradeHistoryInBacktest](https://ninjatrader.com/es/support/helpGuides/nt8/includetradehistoryinbacktest.htm) be set to true. |

**Methods and Properties**

|  |  |
| --- | --- |
| [AllTrades](https://ninjatrader.com/es/support/helpGuides/nt8/alltrades.htm) | Gets a [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) object of all trades generated by the strategy |
| [LongTrades](https://ninjatrader.com/es/support/helpGuides/nt8/longtrades.htm) | Gets a [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) object of long trades generated by the strategy |
| [RealTimeTrades](https://ninjatrader.com/es/support/helpGuides/nt8/realtimetrades.htm) | Gets a [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) object of real-time trades generated by the strategy |
| [ShortTrades](https://ninjatrader.com/es/support/helpGuides/nt8/shorttrades.htm) | Gets a [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) object of short trades generated by the strategy |

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {     // Print out the number of long trades     Print("The strategy has taken " + SystemPerformance.LongTrades.Count + " long trades."); } |

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

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

**In this section**

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

**Definition**

Represents the real-world or simulation **Account** configured for the strategy.

**Property Value**

An [Account](https://ninjatrader.com/es/support/helpGuides/nt8/account_class.htm) object configured for the strategy

**Syntax**

Account

**Examples**

| ns | |
| --- | --- |
| //Displays text on chart indicating what account the strategy is applied to  Draw.TextFixed(this, "tag1", "Strategy is applied to " + Account.Name, 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) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) >  **AddChartIndicator()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/strategy_account.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/addperformancemetric.htm) |

**Definition**

Adds an indicator to the strategy only for the purpose of displaying it on a chart.

|  |
| --- |
| **Notes**:  •Only the Plot properties of an indicator added by AddChartIndicator() will be accessible in the Indicators dialogue on charts. Other properties must be set in code.  •To add Bars objects to your strategy for calculation purposes see the [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) method.  •An indicator being added via AddChartIndicator() cannot use any additional data series hosted by the calling strategy, but can only use the strategy's primary data series. If you wish to use a different data series for the indicator's input, you can add the series in the indicator itself and explicitly reference it in the indicator code (please make sure though the hosting strategy has the same [AddDataSeries()](https://ninjatrader.com/es/support/helpGuides/nt8/adddataseries.htm) call included as well)  o If a secondary or null Bars series is specified by the calling strategy (not the indicator itself), the strategy's primary series will be substituted instead.  •Dynamically using [DrawOnPricePanel](https://ninjatrader.com/es/support/helpGuides/nt8/drawonpricepanel.htm) in an indicator outside of State.SetDefaults may show issues when working with that indicator through a hosting strategy via [AddChartIndicator()](https://ninjatrader.com/es/support/helpGuides/nt8/addchartindicator.htm). |

**Method Return Value**

This method does not return a value.

**Syntax**  
AddChartIndicator(IndicatorBase indicator)

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

**Parameters**

|  |  |
| --- | --- |
| indicator | An indicator object |

**Examples**

| ns |
| --- |
| protected override void OnStateChange() {     if (State == State.DataLoaded)     {         // Charts a 20 period simple moving average to the chart         AddChartIndicator(SMA(20));     } } |

|  |
| --- |
| **Tip**:  If you are adding an indicator which is dependent on the correct [State](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm) of the indicator, you will need to ensure that you are also calling the indicator from the strategy in [OnBarUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onbarupdate.htm), otherwise your indicator will only process in **State.RealTime** for performance optimizations. |

| ns | |
| --- | --- |
| protected override void OnStateChange() {   if (State == State.DataLoaded)   {     // Charts a 20 period simple moving average to the chart     AddChartIndicator(SMA(20));   } }   protected override void OnBarUpdate() {     // call SMA() historically to ensure the indicator processes its historical states as well   double sma = SMA(20)[0]; } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) >  **AddPerformanceMetric()** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/addchartindicator.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/atm_strategy_methods.htm) |

**Definition**

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

**Method Return Value**

This method does not return a value.

**Syntax**  
AddPerformanceMetric(PerformanceMetricBase performanceMetric)

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

**Parameters**

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

**Examples**

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

|  |  |
| --- | --- |
| **Navigation:**  [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) >  **BarsRequiredToTrade** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/getatmstrategyuniqueid.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/barssinceentryexecution.htm) |

**Definition**

The number of historical bars required before the strategy starts processing order methods called in the [OnBarUpdate()](https://ninjatrader.com/es/support/helpGuides/nt8/onbarupdate.htm) method. This property is generally set via the UI when starting a strategy.

|  |
| --- |
| **Note**:  In a multi-series strategy this restriction applies only for the primary Bars object.  This means your can run into situations where the primary bars required to trade have been reached, but the additional bars required have not. Should your strategy logic intertwine calculations across different Bars objects please ensure all Bars objects have met the BarsRequiredToTrade requirement before proceeding. This can be done via checks on the [CurrentBars](https://ninjatrader.com/es/support/helpGuides/nt8/currentbars.htm) array. |

**Property Value**

An int value representing the number of historical bars.  Default value is set to 20.

|  |
| --- |
| **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**

BarsRequiredToTrade

|  |
| --- |
| **Tip**:  When working with a multi-series strategy, real-time bar update events for a particular Bars object are only received when that Bars object has satisfied the BarsRequiredToTrade requirement. To ensure this requirement is met, please use the CurrentBars array. |

**Examples**

| ns **Setting the default BarsRequiredToTrade value** |
| --- |
| protected override void OnStateChange() {     if (State == State.Configure)     {         BarsRequiredToTrade = 20;     } } |

| ns | **Checking BarsRequiredToTrade againt a CurrentBars array** |
| --- | --- |
|  | protected override void OnStateChange() {   if (State == State.SetDefaults)   {     BarsRequiredToTrade = 20;   }   else if (State == State.Configure)   {     // add 30 minute series for calcuation logic     AddDataSeries(BarsPeriodType.Minute, 30);   } }   protected override void OnBarUpdate() {   // do not process order logic until bars required to trade is met   // for both primary and 30-minute series have reached their bars required to trade   if (CurrentBars[0] < BarsRequiredToTrade || CurrentBars[1] < BarsRequiredToTrade)     return;     //order 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) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) >  **CloseStrategy()** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/chartindicators.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/connectionlosshandling.htm) |

**Definition**

Cancels all working orders, closes any existing positions, and finally disables the strategy.  This behavior can also be overridden for a given strategy if desired.

|  |
| --- |
| **Notes**:  •If you choose to override this method using custom logic, the default behavior of the CloseStrategy() method will **NOT** be executed.  For this reason, it is suggested to call the base implementation of CloseStrategy() method within the virtual override to ensure that the strategy is terminated as designed, otherwise it is your responsibility to correctly manage any working orders or positions.  •CloseStrategy() will work of the current strategy position and will not factor in any [StartBehavior](https://ninjatrader.com/es/support/helpGuides/nt8/startbehavior.htm) setting, i.e. calling CloseStrategy() while the script is in a virtual historical position could result in an unwanted position  •The default CloseStrategy() handling will only be applied to the primary series of a MultiSeries NinjaScript strategy. |

**Method Return Value**

This method does not return a value.

**Syntax**  
CloseStrategy(string signalName)

|  |
| --- |
| **Warning**:  This method can only be call before the [State](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm) has reached [State.Terminated](https://ninjatrader.com/es/support/helpGuides/nt8/state.htm) |

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

public override void CloseStrategy(string signalName)  
{

}

**Parameters**

|  |  |
| --- | --- |
| signalName | The signal name which will be used to identify the closing order.  If no signal name exists or is null, "Close" will be substituted instead. |

**Examples**

| ns **Basic usage of CloseStrategy** |
| --- |
| DateTime StartTime = new DateTime();       protected override void OnStateChange() {   if (State == State.SetDefaults)   {                 Name = "ExampleStrategy";     }     else if (State == State.Transition)     StartTime = Core.Globals.Now;       }     protected override void OnBarUpdate() {   // if we're still in position 45 minutes after the start time, close strategy   if(Position.MarketPosition != MarketPosition.Flat && Time[0] >= StartTime.AddMinutes(45))     CloseStrategy("My Custom Close"); } |

| ns **Overriding the Default CloseStrategy logic** | |
| --- | --- |
| public override void CloseStrategy(string signalName) {   Print("Executing Custom Close Logic");     // custom close logic     // call default close action   base.CloseStrategy(signalName); } | |
| **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) >  **IncludeTradeHistoryInBacktest** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/includecommission.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isadoptaccountpositionaware.htm) |

**Definition**

Determines if the strategy will save orders, trades, and execution history. When this property is set to **false** you will see significant memory savings at the expense of having access to the detailed trading information.

|  |
| --- |
| **Notes**:  •Since trade information is not stored you will only see entry/exit executions plotted on the chart with no connecting PnL trade lines.  •This property is set to **true** by default when a strategy is applied to a chart. However, in any other window (Strategies tab, Strategy Analyzer), it will be set to **false** by default. If you are working with a strategy outside of a chart, and would like to save orders, trades, and execution history for reference in your code, you will need to set **IncludeTradeHistoryInBacktest** to **true**in your script.  •During [Strategy Analyzer Optimization](https://ninjatrader.com/es/support/helpGuides/nt8/optimize_a_strategy.htm) in a 32-bit process, the **IncludeTradeHistoryInBacktest** property is forced to **false** due to the limited resources available in a 32-bit environment.  You must use a 64-bit process if trade history is needed during optimization. |

**Property Value**

This property returns **true** if the strategy will include trade history; otherwise, **false**. Default is set to **true**.  Always **false** during a strategy analyzer optimization on a 32-bit process.

|  |
| --- |
| **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**

IncludeTradeHistoryInBacktest

**Examples**

| ns | |
| --- | --- |
| protected override void OnStateChange() {     if (State == State.SetDefaults)     {         // Explicitly include trade history in a backtest         IncludeTradeHistoryInBacktest = true;     } }   protected override void OnBarUpdate() {   // Stop taking trades after 10 trades have been taken since the strategy was enabled   if(SystemPerformance.AllTrades.Count >= 10)       return; } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [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) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) >  **Order** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/optimizationperiod.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/isterminalstate.htm) |

**Definition**

Represents a read only interface that exposes information regarding an order.

•An Order object returned from calling an order method is dynamic in that its properties will always reflect the current state of an order

•The property <Order>.OrderId is **NOT** a unique value, since it can change throughout an order's lifetime.  Please see the [Advance Order Handling](https://ninjatrader.com/es/support/helpGuides/nt8/advanced_order_handling.htm) section on "*Transitioning order references from historical to live"* for details on how to handle.

•The property <Order>.Oco **WILL** be appended with a suffix when the strategy transitions from historical to real-time to ensure the OCO id is unique across multiple strategies for live orders

•To check for equality you can compare Order objects directly

**Methods and Properties**

|  |  |
| --- | --- |
| Account | The [Account](https://ninjatrader.com/es/support/helpGuides/nt8/account_class.htm) the order resides |
| AverageFillPrice | A double value representing the average fill price of an order |
| Filled | An int value representing the filled amount of an order |
| FromEntrySignal | A string representing the user defined fromEntrySignal parameter on an order |
| Gtd | A [DateTime](http://msdn2.microsoft.com/en-us/library/system.datetime.aspx" \t "_blank) structure representing when the order will be canceled |
| HasOverfill | A bool value representing if the order is an overfill. For use when using [Unmanaged orders](https://ninjatrader.com/es/support/helpGuides/nt8/unmanaged_approach.htm) and [IgnoreOverFill](https://ninjatrader.com/es/support/helpGuides/nt8/ignoreoverfill.htm) |
| Instrument | An [Instrument](https://ninjatrader.com/es/support/helpGuides/nt8/instrument.htm) value representing the instrument of an order |
| IsLiveUntilCancelled | A bool when true indicates the order will be canceled by [managed order handling](https://ninjatrader.com/es/support/helpGuides/nt8/managed_approach.htm) at expiration, otherwise false |
| [IsTerminalState()](https://ninjatrader.com/es/support/helpGuides/nt8/isterminalstate.htm) | A static method used to determine if the an order's **OrderState** is in considered terminal and no longer active |
| LimitPrice | A double value representing the limit price of an order |
| Name | A string representing the name of an order which can be provided by the entry or exit signal name |
| Oco | A string representing the OCO (one cancels other) id of an order |
| OrderAction | Represents the action of the order.  Possible values are:  OrderAction.Buy  OrderAction.BuyToCover  OrderAction.Sell  OrderAction.SellShort |
| OrderId | A string representing the broker issued order id value (this value can change) |
| OrderState | The current state of the order.  See the order state values table below |
| OrderType | The type of order submitted.  Possible values are:  OrderType.Limit  OrderType.Market OrderType.MIT  OrderType.StopMarket  OrderType.StopLimit |
| Quantity | An int value representing the quantity of an order |
| StopPrice | A double value representing the stop price of an order |
| Time | A [DateTime](http://msdn2.microsoft.com/en-us/library/system.datetime.aspx" \t "_blank) structure representing the last time the order changed state |
| TimeInForce | Determines the life of the order.  Possible values are:  TimeInForce.Day  TimeInForce.Gtc |
| ToString() | A string representation of an order |

**OrderState Values**

|  |  |
| --- | --- |
| OrderState.Initialized | Order is initialized in NinjaTrader |
| OrderState.Submitted | Order is submitted to the broker |
| OrderState.Accepted | Order is accepted by the broker or exchange |
| OrderState.TriggerPending | Order is pending submission |
| OrderState.Working | Order is working in the exchange queue |
| OrderState.ChangePending | Order change is pending in NinjaTrader |
| OrderState.ChangeSubmitted | Order change is submitted to the broker |
| OrderState.CancelPending | Order cancellation is pending in NinjaTrader |
| OrderState.CancelSubmitted | Order cancellation is submitted to the broker |
| OrderState.Cancelled | Order cancellation is confirmed by the exchange |
| OrderState.Rejected | Order is rejected |
| OrderState.PartFilled | Order is partially filled |
| OrderState.Filled | Order is completely filled |
| OrderState.Unknown | An unknown order state. Default if broker does not report current order state. |

|  |
| --- |
| **Critical**: In a historical backtest, orders will always reach a "Working" state. In real-time, some stop orders may only reach "Accepted" state if they are simulated/held on a brokers server |

**Examples**

|  |  |
| --- | --- |
| ns |  |
| private Order entryOrder = null;   protected override void OnBarUpdate() {   if (entryOrder == null && Close[0] > Open[0])       EnterLong("myEntryOrder"); }   protected override void OnOrderUpdate(Order order, double limitPrice, double stopPrice, int quantity, int filled, double averageFillPrice, OrderState orderState, DateTime time, ErrorCode error, string nativeError) {   // Assign entryOrder in OnOrderUpdate() to ensure the assignment occurs when expected.   // This is more reliable than assigning Order objects in OnBarUpdate, as the assignment is not guaranteed to be complete if it is referenced immediately        after submitting   if (order.Name == "myEntryOrder")       entryOrder = order;     if (entryOrder != null && entryOrder == order)   {       Print(order.ToString());       if (order.OrderState == OrderState.Filled)           entryOrder = null;   } } | |
| **Navigation:**  [NinjaScript](https://ninjatrader.com/es/support/helpGuides/nt8/ninjascript.htm) > [Language Reference](https://ninjatrader.com/es/support/helpGuides/nt8/language_reference_wip.htm) > [Strategy](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) >  **SystemPerformance** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/strategybaseconverter.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/alltrades.htm) |

**Definition**

The SystemPerformance object holds all trades and trade performance data generated by a strategy.

|  |
| --- |
| **Notes**:  •A NinjaScript strategy can generate both synthetic trades (historical backtest trades) and real-time trades executed on a real-time data stream. If you wish to access only real-time trades, access the "RealTimeTrades" collection  •The first trade of the "RealTimeTrades" collection will contain a synthetic entry execution if the strategy was **NOT** flat at the time you start the strategy.  •These properties require that [IncludeTradeHistoryInBacktest](https://ninjatrader.com/es/support/helpGuides/nt8/includetradehistoryinbacktest.htm) be set to true. |

**Methods and Properties**

|  |  |
| --- | --- |
| [AllTrades](https://ninjatrader.com/es/support/helpGuides/nt8/alltrades.htm) | Gets a [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) object of all trades generated by the strategy |
| [LongTrades](https://ninjatrader.com/es/support/helpGuides/nt8/longtrades.htm) | Gets a [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) object of long trades generated by the strategy |
| [RealTimeTrades](https://ninjatrader.com/es/support/helpGuides/nt8/realtimetrades.htm) | Gets a [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) object of real-time trades generated by the strategy |
| [ShortTrades](https://ninjatrader.com/es/support/helpGuides/nt8/shorttrades.htm) | Gets a [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) object of short trades generated by the strategy |

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {     // Print out the number of long trades     Print("The strategy has taken " + SystemPerformance.LongTrades.Count + " long trades."); } |

|  |  |
| --- | --- |
| **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) > [SystemPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/systemperformance.htm) >  **AllTrades** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/systemperformance.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/systemperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/longtrades.htm) |

**Definition**

A [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) object of all trades generated by a strategy.

**Syntax**

SystemPerformance.AllTrades

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the number of long trades     Print("The strategy has taken " + SystemPerformance.AllTrades.Count + " trades."); } | |
| **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) > [SystemPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/systemperformance.htm) >  **LongTrades** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/alltrades.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/systemperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/realtimetrades.htm) |

**Definition**

LongTrades is a [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) object of long trades generated by a strategy.

**Syntax**

SystemPerformance.LongTrades

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the number of long trades     Print("The strategy has taken " + SystemPerformance.LongTrades.Count + " long trades."); } | |
| **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) > [SystemPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/systemperformance.htm) >  **RealTimeTrades** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/longtrades.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/systemperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/shorttrades.htm) |

**Definition**

RealTimeTrades is a [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) object of real-time trades generated by a strategy.

**Syntax**

SystemPerformance.RealTimeTrades

**Examples**

| ns | |
| --- | --- |
| protected override void OnBarUpdate() {     // Print out the number of real-time trades     Print("The strategy has taken " + SystemPerformance.RealTimeTrades.Count + " real-time trades."); } | |
| **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) > [SystemPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/systemperformance.htm) >  **ShortTrades** | | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/realtimetrades.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/systemperformance.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/testperiod.htm) |

**Definition**

ShortTrades is a [TradeCollection](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) object of short trades generated by a strategy.

**Syntax**

SystemPerformance.ShortTrades

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {     // Print out the number of short trades     Print("The strategy has taken " + SystemPerformance.ShortTrades.Count + " short trades."); } |

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

**Definition**

A collection of [Trade](https://ninjatrader.com/es/support/helpGuides/nt8/trade.htm) objects. You can access a trade object by providing an index value. Trades are indexed sequentially meaning the oldest trade taken in a strategy will be at an index value of zero. The most recent trade taken will be at an index value of the total trades in the collection minus 1.

**Methods and Properties**

|  |  |
| --- | --- |
| [TradesCount](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection_tradescount.htm) | An int value representing the number of trades in the collection |
| [EvenTrades](https://ninjatrader.com/es/support/helpGuides/nt8/eventrades.htm) | Gets a TradeCollection object of even trades |
| [GetTrades()](https://ninjatrader.com/es/support/helpGuides/nt8/gettrades.htm) | Gets a TradeCollection object representing a specified position |
| [LosingTrades](https://ninjatrader.com/es/support/helpGuides/nt8/losingtrades.htm) | Gets a TradeCollection object of losing trades |
| [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) | Gets a [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) object |
| [WinningTrades](https://ninjatrader.com/es/support/helpGuides/nt8/winningtrades.htm) | Gets a TradeCollection object of winning trades |

**Examples**

| ns**Example 1** |
| --- |
| protected override void OnBarUpdate() {   // Accesses the first/last trade in the strategy (oldest trade is at index 0)   // and prints out the profit as a percentage to the output window   if (SystemPerformance.AllTrades.Count > 1)   {       Trade lastTrade = SystemPerformance.AllTrades[SystemPerformance.AllTrades.Count - 1];       Trade firstTrade = SystemPerformance.AllTrades[0];         Print("The last trade profit is " + lastTrade.ProfitPercent);       Print("The first trade profit is " + firstTrade.ProfitPercent);   } } |

| ns**Example 2** |
| --- |
| protected override void OnBarUpdate() {   // Once the strategy has executed 20 trades loop through the losing trades   // collection and print out the PnL on only long trades   if (SystemPerformance.AllTrades.Count == 20)   {       Print("There are " + SystemPerformance.AllTrades.LosingTrades.Count + " losing trades.");       foreach (Trade myTrade in SystemPerformance.AllTrades.LosingTrades)       {           if (myTrade.Entry.MarketPosition == MarketPosition.Long)               Print(myTrade.ProfitCurrency);       }   } } |

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

**Definition**

A collection of [Trade](https://ninjatrader.com/es/support/helpGuides/nt8/trade.htm) objects. You can access a trade object by providing an index value. Trades are indexed sequentially meaning the oldest trade taken in a strategy will be at an index value of zero. The most recent trade taken will be at an index value of the total trades in the collection minus 1.

**Methods and Properties**

|  |  |
| --- | --- |
| [TradesCount](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection_tradescount.htm) | An int value representing the number of trades in the collection |
| [EvenTrades](https://ninjatrader.com/es/support/helpGuides/nt8/eventrades.htm) | Gets a TradeCollection object of even trades |
| [GetTrades()](https://ninjatrader.com/es/support/helpGuides/nt8/gettrades.htm) | Gets a TradeCollection object representing a specified position |
| [LosingTrades](https://ninjatrader.com/es/support/helpGuides/nt8/losingtrades.htm) | Gets a TradeCollection object of losing trades |
| [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) | Gets a [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) object |
| [WinningTrades](https://ninjatrader.com/es/support/helpGuides/nt8/winningtrades.htm) | Gets a TradeCollection object of winning trades |

**Examples**

| ns**Example 1** |
| --- |
| protected override void OnBarUpdate() {   // Accesses the first/last trade in the strategy (oldest trade is at index 0)   // and prints out the profit as a percentage to the output window   if (SystemPerformance.AllTrades.Count > 1)   {       Trade lastTrade = SystemPerformance.AllTrades[SystemPerformance.AllTrades.Count - 1];       Trade firstTrade = SystemPerformance.AllTrades[0];         Print("The last trade profit is " + lastTrade.ProfitPercent);       Print("The first trade profit is " + firstTrade.ProfitPercent);   } } |

| ns**Example 2** |
| --- |
| protected override void OnBarUpdate() {   // Once the strategy has executed 20 trades loop through the losing trades   // collection and print out the PnL on only long trades   if (SystemPerformance.AllTrades.Count == 20)   {       Print("There are " + SystemPerformance.AllTrades.LosingTrades.Count + " losing trades.");       foreach (Trade myTrade in SystemPerformance.AllTrades.LosingTrades)       {           if (myTrade.Entry.MarketPosition == MarketPosition.Long)               Print(myTrade.ProfitCurrency);       }   } } |

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

**Definition**

A collection of [Trade](https://ninjatrader.com/es/support/helpGuides/nt8/trade.htm) objects. You can access a trade object by providing an index value. Trades are indexed sequentially meaning the oldest trade taken in a strategy will be at an index value of zero. The most recent trade taken will be at an index value of the total trades in the collection minus 1.

**Methods and Properties**

|  |  |
| --- | --- |
| [TradesCount](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection_tradescount.htm) | An int value representing the number of trades in the collection |
| [EvenTrades](https://ninjatrader.com/es/support/helpGuides/nt8/eventrades.htm) | Gets a TradeCollection object of even trades |
| [GetTrades()](https://ninjatrader.com/es/support/helpGuides/nt8/gettrades.htm) | Gets a TradeCollection object representing a specified position |
| [LosingTrades](https://ninjatrader.com/es/support/helpGuides/nt8/losingtrades.htm) | Gets a TradeCollection object of losing trades |
| [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) | Gets a [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) object |
| [WinningTrades](https://ninjatrader.com/es/support/helpGuides/nt8/winningtrades.htm) | Gets a TradeCollection object of winning trades |

**Examples**

| ns**Example 1** |
| --- |
| protected override void OnBarUpdate() {   // Accesses the first/last trade in the strategy (oldest trade is at index 0)   // and prints out the profit as a percentage to the output window   if (SystemPerformance.AllTrades.Count > 1)   {       Trade lastTrade = SystemPerformance.AllTrades[SystemPerformance.AllTrades.Count - 1];       Trade firstTrade = SystemPerformance.AllTrades[0];         Print("The last trade profit is " + lastTrade.ProfitPercent);       Print("The first trade profit is " + firstTrade.ProfitPercent);   } } |

| ns**Example 2** |
| --- |
| protected override void OnBarUpdate() {   // Once the strategy has executed 20 trades loop through the losing trades   // collection and print out the PnL on only long trades   if (SystemPerformance.AllTrades.Count == 20)   {       Print("There are " + SystemPerformance.AllTrades.LosingTrades.Count + " losing trades.");       foreach (Trade myTrade in SystemPerformance.AllTrades.LosingTrades)       {           if (myTrade.Entry.MarketPosition == MarketPosition.Long)               Print(myTrade.ProfitCurrency);       }   } } |

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

**Definition**

A collection of [Trade](https://ninjatrader.com/es/support/helpGuides/nt8/trade.htm) objects. You can access a trade object by providing an index value. Trades are indexed sequentially meaning the oldest trade taken in a strategy will be at an index value of zero. The most recent trade taken will be at an index value of the total trades in the collection minus 1.

**Methods and Properties**

|  |  |
| --- | --- |
| [TradesCount](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection_tradescount.htm) | An int value representing the number of trades in the collection |
| [EvenTrades](https://ninjatrader.com/es/support/helpGuides/nt8/eventrades.htm) | Gets a TradeCollection object of even trades |
| [GetTrades()](https://ninjatrader.com/es/support/helpGuides/nt8/gettrades.htm) | Gets a TradeCollection object representing a specified position |
| [LosingTrades](https://ninjatrader.com/es/support/helpGuides/nt8/losingtrades.htm) | Gets a TradeCollection object of losing trades |
| [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) | Gets a [TradesPerformance](https://ninjatrader.com/es/support/helpGuides/nt8/tradesperformance.htm) object |
| [WinningTrades](https://ninjatrader.com/es/support/helpGuides/nt8/winningtrades.htm) | Gets a TradeCollection object of winning trades |

**Examples**

| ns**Example 1** |
| --- |
| protected override void OnBarUpdate() {   // Accesses the first/last trade in the strategy (oldest trade is at index 0)   // and prints out the profit as a percentage to the output window   if (SystemPerformance.AllTrades.Count > 1)   {       Trade lastTrade = SystemPerformance.AllTrades[SystemPerformance.AllTrades.Count - 1];       Trade firstTrade = SystemPerformance.AllTrades[0];         Print("The last trade profit is " + lastTrade.ProfitPercent);       Print("The first trade profit is " + firstTrade.ProfitPercent);   } } |

| ns**Example 2** |
| --- |
| protected override void OnBarUpdate() {   // Once the strategy has executed 20 trades loop through the losing trades   // collection and print out the PnL on only long trades   if (SystemPerformance.AllTrades.Count == 20)   {       Print("There are " + SystemPerformance.AllTrades.LosingTrades.Count + " losing trades.");       foreach (Trade myTrade in SystemPerformance.AllTrades.LosingTrades)       {           if (myTrade.Entry.MarketPosition == MarketPosition.Long)               Print(myTrade.ProfitCurrency);       }   } } |

|  |  |
| --- | --- |
| **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) >  **Trade** | [Previous page](https://ninjatrader.com/es/support/helpGuides/nt8/traceorders.htm) [Return to chapter overview](https://ninjatrader.com/es/support/helpGuides/nt8/strategy.htm) [Next page](https://ninjatrader.com/es/support/helpGuides/nt8/tradecollection.htm) |

**Definition**

A Trade is a completed buy/sell or sell/buy transaction. It consists of an entry and exit execution.

|  |  |
| --- | --- |
| **Example 1** | **Example 2** |
| Buy 1 contract at a price of 1000 and sell 1 contract at a price of 1001 is one complete trade. | Buy 2 contracts at a price of 1000 and sell the 1st contract at a price of 1001, then sell the 2nd contract at a price of 1002 are **two** completed trades. |

In the second example above, two trade objects are created to represent each individual trade. Each trade object will hold the **same** entry execution for two contracts since this single execution was the opening execution for both individual trades.

**Methods and Properties**

|  |  |
| --- | --- |
| Commission | A double value representing the commission of the trade |
| Entry | Gets an [Execution](https://ninjatrader.com/es/support/helpGuides/nt8/execution.htm) object representing the entry |
| EntryEfficiency | A double value representing the entry efficiency of the trade |
| Exit | Gets an [Execution](https://ninjatrader.com/es/support/helpGuides/nt8/execution.htm) object representing the exit |
| ExitEfficiency | A double value representing the exit efficiency of the trade |
| MaeCurrency | A double value representing max adverse excursion in currency |
| MaePercent | A double value representing max adverse excursion as a percentage |
| MaePips | A double value representing max adverse excursion in pips |
| MaePoints | A double value representing max adverse excursion in points |
| MaeTicks | A double value representing max adverse excursion in ticks |
| MfeCurrency | A double value representing max favorable excursion in currency |
| MfePercent | A double value representing max favorable excursion as a percentage |
| MfePips | A double value representing max favorable excursion in pips |
| MfePoints | A double value representing max favorable excursion in points |
| MfeTicks | A double value representing max favorable excursion in ticks |
| ProfitCurrency | A double value representing profit quoted in currency. |
| ProfitPercent | A double value representing profit as a percentage |
| ProfitPips | A double value representing profit in pips |
| ProfitPoints | A double value representing profit in points |
| ProfitTicks | A double value representing profit in ticks |
| Quantity | An int value representing the quantity of the trade |
| TotalEfficiency | A double value representing the total efficiency of the trade |
| TradeNumber | An int value representing the trade numbed by the sequence it occurred |
| ToString() | A string representation of the Trade object |

**Examples**

| ns |
| --- |
| protected override void OnBarUpdate() {   if (SystemPerformance.RealTimeTrades.Count > 0)   {       // Check to make sure there is at least one trade in the collection       Trade lastTrade = SystemPerformance.RealTimeTrades[SystemPerformance.RealTimeTrades.Count - 1];         // Calculate the PnL for the last completed real-time trade       double lastProfitCurrency = lastTrade.ProfitCurrency;         // Store the quantity of the last completed real-time trade       double lastTradeQty = lastTrade.Quantity;         // Pring the PnL to the NinjaScript Output window       Print("The last trade's profit in currency is " + lastProfitCurrency);       // The trade profit is quantity aware, we can easily print the profit per traded unit as well       Print("The last trade's profit in currency per traded unit is " + (lastProfitCurrency / lastTradeQty));   } } |