

Automated Trading Management

- [Introduction](#)
- [Variables](#)
 - [AllowMultipleEntriesInSameDirection](#)
 - [MaximumPositionAllowed](#)
 - [SupportReversals](#)
 - [SendOrdersToTradeService](#)
 - [AllowOppositeEntryWithOpposingPositionOrOrders](#)
 - [SupportAttachedOrdersForTrading](#)
 - [CancelAllOrdersOnEntriesAndReversals](#)
 - [CancelAllOrdersOnReversals](#)
 - [CancelAllOrdersOnEntries](#)
 - [AllowEntryWithWorkingOrders](#)
 - [CancelAllWorkingOrdersOnExit](#)
 - [SupportTradingScaleIn](#)
 - [SupportTradingScaleOut](#)
- [Unmanaged Automated Trading](#)
- [Order Actions](#)
 - [Buy Entry](#)
 - [Buy Exit](#)
 - [Sell Entry](#)
 - [Sell Exit](#)
- [Using Multiple Automated Trading Systems at the Same Time](#)
- [Differences Between Live Automated Trading and Simulated Automated Trading](#)
- [Automatic Recovery of Trading System After Restart](#)
- [Troubleshooting Automated Trading System Behavior](#)
- [Understanding Calculation of Trading System Studies](#)
- [Low Latency Automated Trading](#)
- [Order of Execution](#)

Introduction

This documentation applies to version 1355 and higher. Since version 1355, the following change has been made:

- Market orders are not attempted to be canceled when orders are set to be canceled on entries, reversals, and exits because market orders cannot be expected to be canceled. Therefore, market orders are considered in calculations of working orders when combined with the current Trade Position Quantity, even if the variables to cancel orders have been set to TRUE/Yes.

Sierra Chart supports automated trading. There are three methods to perform automated trading. They are as follows.

1. The easiest and most interactive way to do automated trading is by using the **Spreadsheet System for Trading** study. For complete step-by-step instructions and an example, refer to the [Spreadsheet System for Trading](#) study page. This is a good way to start with automated trading.
2. For a more advanced way to do automated trading that gives you a higher degree of control and maximum performance, you will want to use the [Advanced Custom Study/System Interface and Language](#) and use the [ACSIL Trading Functions](#). These links refer to the necessary documentation. There are many examples provided with Sierra Chart.
3. A very simplified method of automated trading is to use the [Trading System Based on Alert Condition](#) study.

In Sierra Chart, automated trading is performed through studies that function as automated trading systems. You can use either the [Spreadsheet System for Trading](#) study or the [ACSIL Trading Functions](#) in an Advanced Custom Study that you write. These studies will be applied to charts that you have open.

There is no limit to the number of automated trading systems that can be running at the same time. However, having more than one automated trading system for the same symbol at the same time can only be done if each chart that has the automated trading system study is using a different Trade Account. Refer to [Selecting Trade Account](#) on the Trade Window. However, it still might be possible if using the same Trade Account. For more information, refer to [Using Multiple Automated Trading Systems at the Same Time](#).

With either method of automated trading, this page explains all of the automated trading Variables and Order Actions. These Variables and Order Actions along with the functionality provided by the **Spreadsheet System for Trading** study or the **ACSIL Trading** functions, provide the automated trading functionality in Sierra Chart.

To back test an automated trading system and view trading results, refer to the [Auto Trade System Back Testing](#) page. Developing an automated trading system and back testing it, is the only method by which you can determine the profitability of a mechanical trading method.

Variables

These auto trade variables apply to both the **Advanced Custom Study Interface and Language (ACSIL)** (in this case they must be prefixed with **sc.**) and they also apply to the **Spreadsheet System for Trading** study (in this case they are study Inputs).

When a reference is made to the Trade Position, this refers to the current Trade Position for the symbol of the chart that the trading system is applied to.

When a reference is made to working orders, this refers to orders for the symbol of the chart that the trading system is applied to with a **Status**, as displayed in the **Trade >> Trade Orders Window**, of **Pending Open, Open, Pending Cancel, or Pending Modify**. **Pending Child** orders are ignored.

An order can either be simulated or non-simulated. If the automated trading system is in simulated mode, then the reference to working orders is for simulated working orders. If the automated trading system is a non-simulated mode, then the reference is to non-simulated working orders.

AllowMultipleEntriesInSameDirection

The default value for this variable is **FALSE (No)**. When this variable is set to **FALSE (No)**, the Automated Trading Management system will not allow multiple Entry orders in the same direction. More details about how this works can be found in the [BuyEntry](#) and [SellEntry](#) Order Action sections.

If you want to allow Multiple Entries in the Same Direction, set this variable to TRUE (Yes).

MaximumPositionAllowed

The information below applies to version 1599 and higher.

MaximumPositionAllowed controls the maximum Trade Position Quantity allowed when performing automated trading for a particular Symbol and Trade Account. An order that would increase the Trade Position for the Symbol and Trade Account to a quantity greater than the specified **Maximum Position Allowed** will be ignored/rejected.

For example, if this variable is set to 5, a Buy order that would increase the existing Trade Position Quantity greater than 5 will be ignored. And a Sell order that would increase the existing Trade Position Quantity less than -5 will be ignored.

The following explains how the Maximum Position Allowed is enforced:

When an order signal is given, and [CancelAllOrdersOnEntriesAndReversals](#) is FALSE, then the quantities of existing working orders are added with the current Trade Position Quantity.

If the current Trade Position Quantity is positive/long, then the quantity of sell working orders which will reduce/exit this Position Quantity are not included in this calculation unless they are market orders.

If the current Trade Position Quantity is negative/short, then the quantity of buy working orders which will reduce/exit this Position Quantity are not included in this calculation unless they are market orders.

The quantity of the new order submitted by the automated trading system is added to the calculated sum. If this final sum exceeds the Maximum Position Allowed, then the order signal will be ignored.

When calculating the quantities of orders, Buy orders are considered positive and Sell orders are considered negative.

When an order signal is given, and [CancelAllOrdersOnEntriesAndReversals](#) is TRUE, then the quantities of existing working market orders are added with the current Trade Position Quantity. The quantity of the new order submitted by the automated trading system is added to this sum. If this final sum exceeds in the Maximum Position Allowed, then the order signal will be ignored.

It is possible there can be certain order configurations which could workaround the Maximum Allowed Position for automated trading. This could be true in the case of when using Limit orders which do not immediately fill. Alternatively, there is also the Trade Position Limit which can be set for an individual symbol (not the global position limit), which uses a more comprehensive method to limit the Position Quantity. Refer to [Global Symbol Settings](#).

[CancelAllOrdersOnEntriesAndReversals](#) does not cancel market orders.

For example, if there is a Trade Position Quantity of +1 and an existing Buy order of quantity 1, then this is a potential Position of 2. If the **Maximum Position Allowed** is 2 and you submit another Buy order, it will not be allowed.

When calculating the quantity of working orders, orders that are in an OCO group are counted as 1 order. An OCO group contains two orders of the same side either both buy or both sell.

It needs to be understood that working non-market orders that are attempted to be canceled when using [CancelAllOrdersOnEntriesAndReversals](#) may not actually be canceled and can be filled and this can lead to the Maximum Position Allowed being exceeded.

When determining the potential Trade Position Quantity from working orders, you also need to consider how both Buy and Sell orders that are working will oppose each other in quantities. For example, a Buy order with a quantity of 3 and a Sell order with a quantity of 3, will be considered to have a zero potential effect on the Trade Position Quantity if it is currently 0. If the Buy order fills and the Sell order does not, then the Position will be 3. Therefore, it is possible under this kind of condition that no matter what you have set the [MaximumPositionAllowed](#) to, that you still can end up exceeding this setting.

SupportReversals

The default value for the **SupportReversals** variable is **FALSE (No)**. If this variable is set to **TRUE (Yes)**, then Trade Position reversals are supported.

A reversal is triggered when a [Buy Entry](#) or [Sell Entry](#) order is submitted and an actual opposite Trade Position currently exists for the symbol. If there is not an actual Trade Position for the symbol, then a reversal cannot happen.

When a reversal order is given it will flatten the existing Position, and create a new Position which will have a quantity equal to the **Order Quantity** specified. Therefore, there must be an Order Quantity specified.

When calculating the Order Quantity for flattening the existing Trade Position, the Auto Trade Management system looks at the current Trade Position quantity and the quantities of working orders that can exit the Position only.

It does not consider working orders that can increase the size of the Position. For example, if the Position is +5 and there is a [Sell Entry](#), and there is a working sell order with a quantity of 2, then the order to flatten the Position will be a Sell order of quantity 3. Additionally, if those working orders already indicate the potential Position is flat or does not exist, then no reversal actually occurs, only a new entry.

If [CancelAllOrdersOnEntriesAndReversals](#) is set to TRUE (Yes), then only the actual Trade Position quantity and working market orders that will reduce the Position are considered when determining the quantity to flatten the current Trade Position. There is no consideration of other working orders because they will be canceled. If the working market orders will flatten the Position, then no reversal actually occurs.

An example of [SupportReversals](#) is as follows: The current Trade Position quantity is +3 (long Position of quantity 3), there is a [Sell Entry](#) order action with a quantity of 1, the Auto Trade Management system will send a Sell order with a quantity that will create a resulting Position of -1. This will be a quantity of 4.

Note, that when [SupportReversals](#) is set to **TRUE (Yes)**, then the value of the [AllowOppositeEntryWithOpposingPositionOrOrders](#) variable is ignored.

If a reversal occurs, a message will be added to the **Trade >> Trade Service Log** indicating that an automated trading reversal has occurred. The chart and the trading study will be given in the message.

If [CancelAllOrdersOnEntriesAndReversals](#) is set to **TRUE (Yes)**, then any Attached Orders for the existing Trade Position will be immediately canceled.

Actual Order Types and Quantities Submitted

The above documentation for [SupportReversals](#) describes the logical Order and quantity of that Order which is submitted.

When SupportReversals causes a reversal order to be submitted, then there are actually 2 orders submitted.

There is the order to flatten the current Trade Position and it is a Market order.

There is the order which establishes a new Trade Position and will be the order type which is specified by the ACSIL Trading system or the Spreadsheet Trading system. However, the order which flattens the current Trade Position will always be a Market order.

The reason for this design is in support using a different order type like a Limit order when a reversal is triggered. So that the new Trade Position will be established with this order type and so that the current Trade Position will be immediately flattened.

Whether 2 orders are always required in all cases does not matter. There are two orders for implementation consistency and consistency of this functionality.

Also when using Attached Orders it is necessary that the order to flatten the Trade Position is separate from the order to establish a new Trade Position because the order to create the new Position must be matched in quantity to its Attached Orders and the Attached Orders must be matched in quantity to the quantity of the new Position.

Therefore, if you want to use a single order when reversing a Trade Position, then do not use the

[SupportReversals](#) automated trading flag in your automated trading system and instead use [BuyEntry](#) or [SellEntry](#) and specify the quantity required to reverse the current Trade Position.

SupportReversals in no way has any effect on [Buy Exit](#) or [Sell Exit](#). Those still function as they normally do.

SendOrdersToTradeService

This variable applies to the automated trading system it is set on. The default value for this variable is FALSE (No).

Setting **SendOrdersToTradeService** to FALSE (No) will send all orders and order actions to the Sierra Chart Trade Simulation system.

Setting **SendOrdersToTradeService** to TRUE (Yes) will send all orders and order actions to the connected Trading service. The connected Trading service is defined in the [Data/Trade Service Settings](#).

However, the **SendOrdersToTradeService** automated trading management variable must be set consistently with the global **Trade >> Trade Simulation Mode On** setting on the menu, otherwise order actions will be ignored.

If global Trade Simulation Mode is OFF (unchecked), then **SendOrdersToTradeService** must be set to Yes/TRUE for any order action (submitting, modifying, canceling order) to work, otherwise they will be ignored and there will be an error given when these actions occur. There will also be a message added to the **Trade >> Trade Service Log** like this: **[Chart Name]: SendOrdersToTradeService is not consistent with global Trade Simulation Mode setting. Order action ignored.**

If global Trade Simulation Mode is ON, then **SendOrdersToTradeService** must be set to No/FALSE for any order action (submitting, modifying, canceling order) to work, otherwise they will be ignored and there will be an error given when these order actions occur. There will also be a message added to the **Trade >> Trade Service Log**.

Even though an order is sent to your Trading service by disabling Trade Simulation Mode and by setting this variable to TRUE (Yes), you may be connected to a simulated account with your Trading service. In this case the trades will still be simulated.

SendOrdersToTradeService applies to each individual trading system that you have. It is not a global setting and is separate from the global Trade Simulation Mode setting (**Trade >> Trade Simulation Mode On**).

In the case of when using ACSIL, if using a study [Input](#) to control **sc.SendOrdersToTradeService**, then it is necessary to change the value of this variable after the **sc.SetDefaults** code block. You cannot do this within that code block because changes to the Input will have no effect.

AllowOppositeEntryWithOpposingPositionOrOrders

The default value for this variable is **TRUE (Yes)**.

When this variable is set to **FALSE (No)**, then a [Buy Entry](#) or [Sell Entry](#) is not allowed if there is a Trade Position or working orders already existing in the opposite direction. Otherwise, the signal will be allowed.

Complete details about how this variable works can be found in the [BuyEntry](#) and [SellEntry](#) Order Action sections.

If this is set to TRUE (Yes), you should set [CancelAllOrdersOnEntriesAndReversals](#) to TRUE (Yes) when using Attached Orders so that those Attached Orders will be canceled since the current Trade Position will be reduced in quantity or reversed.

SupportAttachedOrdersForTrading

It is not necessary to use the [BuyExit](#) or [SellExit](#) Order Actions to exit a Position. Exits can be performed by using the Attached Orders feature. Attached Orders are Target and Stop orders attached to the primary/parent order. These orders are managed by Sierra Chart and are sent to the trading service when the parent order fills. Although, is completely fine to use [BuyExit](#) or [SellExit](#) and not Attached Orders. It is possible to use both Attached Orders and [BuyExit](#) or [SellExit](#) to exit the Position. This is explained in this section.

Targets and Stops are configured on the **Attached Orders** tab of the Trade Window associated with the chart that the trading study is applied to. For more information about configuring Attached Orders through the Trade Window, refer to the [Attached Orders](#) documentation page. Various order types for Attached Orders are supported including trailing stops. In the case of an ACSIL trading system, it can programmatically specify the Attached Orders or specify a Trade Window configuration file (explained below).

The default value for this variable is **FALSE (No)** for **ACSIL** trading system studies. In the case of ACSIL trading system studies, when this variable is TRUE, then the configured Attached Orders on the Trade Window will be used.

For the **Spreadsheet System for Trading** study, this variable is **TRUE (Yes)** and cannot be changed. However, for Attached Orders to actually be used in the case of Spreadsheet trading, the **Use Attached Orders** option must also be enabled on the Trade Window for the chart.

When this variable is set to **TRUE (Yes)**, the [BuyEntry](#) and the [SellEntry](#) Order Actions will use the **Attached Orders** that are specified on the associated **Trade Window** for the chart that your trading study is applied to. This only applies to the [BuyEntry](#) and [SellEntry](#) Order Actions. [BuyExit](#) or [SellExit](#) Order Actions cannot have Attached Orders attached to them.

It is possible to use both Attached Orders and [BuyExit](#) and [SellExit](#). For example, you may want to use a fixed Target, however you may want to exit your Position using dynamic rules from your trading system.

In order to use [BuyExit](#) and [SellExit](#) with Attached Orders that are configured to flatten your Position, you must set [CancelAllWorkingOrdersOnExit](#) to TRUE (Yes). This allows you to use both. If the Attached Orders are not set to flatten your Position completely, then [BuyExit](#) and [SellExit](#) can be used without setting [CancelAllWorkingOrdersOnExit](#) to TRUE. In this case when [BuyExit](#) or [SellExit](#)

are triggered they will close the remaining Position that the Attached Orders are not set to. It is also possible to use the [Scale Out](#) functionality with [BuyExit](#) or [SellExit](#), to reduce the quantities of Attached Orders that are working and in this case keep [CancelAllWorkingOrdersOnExit](#) set to FALSE.

A set of Attached Orders, which is entered on the Trade Window for a chart, is saved when a chartbook is saved. The Trade Window configuration, which also includes Attached Orders can also be saved to a Trade Window configuration file (.twconfig file extension). To save a configuration file, select the **Menu/M** menu on the Trade Window and select the **Save Configuration** command.

Programmatically Specifying A Trade Window Configuration File For Attached Orders: If your automated trading system uses Attached Orders, it is a good idea to have saved the [Trade Window Configuration](#) to a file that contains the Attached Orders defined on the Trade Window. To specify this Trade Window Configuration file in your ACSIL Trading system, use this example:

Example

```
// This line can be within sc.SetDefaults code block in your trading system function or outside of it.  
sc.TradeWindowConfigFileName = "SimpleBracket.twconfig";
```

Any time a ACSIL trading function is used, the Trade Window Configuration will be what has been last specified by the **sc.TradeWindowConfigFileName** text string variable. So this can be set right before an ACSIL trading function is used.

To use a set of Attached Orders for an ACSIL trading system and not have to rely on a Chartbook or a Trade Window configuration file, refer to [these members of the s_SCNewOrder structure](#). For a complete code example, refer to the **scsf_TradingExampleWithAttachedOrdersDirectlyDefined** function in the **TradingSystem.cpp** file in the **/ACS_Source** folder in the folder where Sierra Chart is installed to.

Note: It definitely is not recommended for you to implement yourself in an ACSIL trading system, Target and Stop orders by sending both of them after the parent order fills, and handling the related OCO functionality. The reason for this, is there is a very high degree of sophistication involved to actually implement OCO functionality properly. And it is something that Sierra Chart internally will manage for you through the Attached Orders feature. Sierra Chart will properly manage this for you in a reliable way in a live trading environment.

CancelAllOrdersOnEntriesAndReversals

When **CancelAllOrdersOnEntriesAndReversals** is set to TRUE (Yes), and there is a [Buy Entry](#) or a [Sell Entry](#) or a Reversal Order Action given, then all working orders except for market orders, for the Symbol and Trade Account of the chart that the trading system study is applied to will be canceled before the entry or reversal order is submitted.

The entry or reversal order is submitted immediately after the orders are requested to be canceled. If the order or orders cannot be canceled and the order cancellation has been rejected by the Trading

service, then this can result in a Trade Position Quantity which may exceed what was intended. This is something to consider.

When **CancelAllOrdersOnEntriesAndReversals** is set to FALSE (No), then existing working orders for the Symbol and Trade Account will not be canceled.

CancelAllOrdersOnReversals

When **CancelAllOrdersOnReversals** is set to TRUE (Yes), and there is a [Buy Entry](#) or a [Sell Entry](#) and which also is reversing the existing Trade Position, then all working orders except for market orders, for the Symbol and Trade Account of the chart that the trading system study is applied to will be canceled before the reversal order is submitted.

The reversal order is submitted immediately after the orders are requested to be canceled. If the order or orders cannot be canceled and the order cancellation has been rejected by the Trading service, then this can result in a Trade Position Quantity which may exceed what was intended. This is something to consider.

When **CancelAllOrdersOnReversals** is set to FALSE (No), then existing working orders for the Symbol and Trade Account will not be canceled.

CancelAllOrdersOnEntries

When **CancelAllOrdersOnEntries** is set to TRUE (Yes), and there is a [Buy Entry](#) or a [Sell Entry](#), then all working orders except for market orders, for the Symbol and Trade Account of the chart that the trading system study is applied to will be canceled before the entry order is submitted.

The entry order is submitted immediately after the orders are requested to be canceled. If the order or orders cannot be canceled and the order cancellation has been rejected by the Trading service, then this can result in a Trade Position Quantity which may exceed what was intended. This is something to consider.

When **CancelAllOrdersOnEntries** is set to FALSE (No), then existing working orders for the Symbol and Trade Account will not be canceled.

AllowEntryWithWorkingOrders

The default for this variable is FALSE (No). When **AllowEntryWithWorkingOrders** is set to TRUE (Yes), then a [Buy Entry](#) or a [Sell Entry](#) will be allowed if there are existing working orders of any type.

There can still be other exceptions to prevent an Entry order from being processed. When **AllowEntryWithWorkingOrders** is set to FALSE (No), then if there are existing working orders, then a [Buy Entry](#) or a [Sell Entry](#) Order Action, will not be allowed even if [CancelAllOrdersOnEntriesAndReversals](#) is set to TRUE.

When **AllowEntryWithWorkingOrders** is set to FALSE (No), then when an order is not allowed, a message will be added to **Trade >> Trade Service Log**.

CancelAllWorkingOrdersOnExit

The default for this variable is FALSE (No). When **CancelAllWorkingOrdersOnExit** is set to TRUE (Yes), and the [BuyExit](#) or [SellExit](#) Order Actions are used, working orders, except market orders, that can exit the Position will not be used in the calculation to determine the current Trade Position Quantity with working orders. Only the actual Trade Position Quantity and Market orders which will exit the Trade Position will be evaluated to determine if an exit is possible.

In this case, if an exit is possible, then all working orders except market orders will be canceled and the current Trade Position will be flattened or reduced by the specified quantity.

One useful purpose for this variable is that it allows you to use **Attached Orders** and also use the [BuyExit](#) and [SellExit](#) Order Actions at the same time as Attached Orders. Therefore, when you use the [BuyExit](#) or [SellExit](#) Order Actions, the working Attached Orders will be canceled.

CancelAllWorkingOrdersOnExit cancels working orders except market orders at the time the [BuyExit](#) and [SellExit](#) Order Actions are used. This does not happen upon the order filling, but at the time the order is submitted.

SupportTradingScaleIn

The **SupportTradingScaleIn** variable relates to Attached Orders. For a description of **Scaling In** functionality, refer to [Scaling In](#). It can only be set programmatically within an ACSIL trading system. The specific variable is **sc.SupportTradingScaleIn** and must be set outside of the **sc.SetDefaults** code block.

SupportTradingScaleOut

The **SupportTradingScaleOut** variable relates to Attached Orders and other OCO orders. For a description of **Scaling Out** functionality, refer to [Scaling Out](#). It can only be set programmatically within an ACSIL trading system. The specific variable is **sc.SupportTradingScaleOut** and must be set outside of the **sc.SetDefaults** code block.

Unmanaged Automated Trading

If you want to enter **Buy** and **Sell** orders without any restriction and not use the automated trading management logic provided by Sierra Chart, then set the following variables as shown:

- [AllowMultipleEntriesInSameDirection](#) to **TRUE (Yes)**
- [AllowOppositeEntryWithOpposingPositionOrOrders](#) to **TRUE (Yes)**
- [CancelAllOrdersOnEntriesAndReversals](#) to **FALSE (No)**
- [CancelAllWorkingOrdersOnExit](#) to **FALSE (No)**
- [AllowEntryWithWorkingOrders](#) to **TRUE (Yes)**
- [sc.AllowOnlyOneTradePerBar](#) to **FALSE** (applies only to ACSIL trading systems)
- [MaximumPositionAllowed](#) to the maximum Trade Position Quantity you want to allow for your automated trading system or set this to twice that Quantity to ensure

MaximumPositionAllowed does not restrict order entry.

By setting these automated trading variables as shown above, this will give you unrestricted functionality. You will then be able to use the [BuyEntry](#) and the [SellEntry](#) Order Actions to freely enter Buy and Sell orders respectively. In ACSIL you can use the [sc.BuyOrder\(\)](#) and [sc.SellOrder\(\)](#) functions which are equivalent to **sc.BuyEntry()** and **sc.SellEntry()**.

With unmanaged automated trading, the ACSIL **sc.BuyEntry()** and **sc.SellEntry()** functions are not necessarily used for entries. Therefore, Sierra Chart provides the **sc.BuyOrder()** and **sc.SellOrder()** ACSIL functions with more logical naming for Unmanaged Automated Trading. However, no matter which set of functions you use, they are the same and unrestricted based upon the above settings.

If you wish to flatten or reduce the current Trade Position you will need to use the [BuyEntry](#) and the [SellEntry](#) Order Actions.

You could also use the [BuyExit](#) or [SellExit](#) Order Actions as well to flatten or reduce the current Trade Position, however when a study is configured for Unmanaged Automated Trading, you will still have restrictions when using the [BuyExit](#) or [SellExit](#) Order Actions. Therefore, to not have any restrictions, use only the **BuyEntry** | **BuyOrder** and the **SellEntry** | **SellOrder** Order Actions.

These functions can be used whether you are establishing a new Trade Position or flattening or reducing a Trade Position. It does not matter.

In the case of when trading a different Symbol and/or Trade Account as explained in the [Submitting and Managing Orders for Different Symbol and/or Trade Account](#) section by specifying a different Symbol or Trade Account in the s_SCNewOrder structure, the trading is always unmanaged.

Order Actions

This section explains how the 4 Order Actions are managed by the Auto Trade Management system in Sierra Chart.

- [Buy Entry](#)
- [Buy Exit](#)
- [Sell Entry](#)
- [Sell Exit](#)

Buy Entry

When the Buy Entry Order Action is triggered, a Buy order will be sent to your connected trading service or to the Trade Simulation System, with the specified quantity as long as certain exceptions are not met.

These exceptions do not apply when using [Unmanaged Automated Trading](#).

The following are the exceptions when using managed automated trading (the default):

1. When [AllowMultipleEntriesInSameDirection](#) is set to FALSE (No), and the current Trade Position Quantity is greater than 0, then the Order Action will not

be allowed.

Or when [AllowMultipleEntriesInSameDirection](#) is set to FALSE (No), [CancelAllOrdersOnEntriesAndReversals](#) is set to FALSE (No), and the current Trade Position Quantity combined with the quantities of all working orders is greater than 0, then the Order Action will not be allowed.

Or when [AllowMultipleEntriesInSameDirection](#) is set to FALSE (No), [CancelAllOrdersOnEntriesAndReversals](#) is set to TRUE (Yes) and the Current Trade Position Quantity combined with the quantity of working market orders is greater than 0, then the Order Action will not be allowed.

Note: When the quantities of working orders are added together and combined with the Trade Position Quantity there are some special rules in the calculations to be aware of. Pending Child orders are ignored. In the case of Target and Stop orders, which are orders in an OCO Group, only the quantity of one of these is used, since one will cancel the other.

2. If [AllowEntryWithWorkingOrders](#) is set to FALSE (No) and there are working orders existing for the symbol, then the Order Action will not be allowed.
3. If **AllowOppositeEntryWithOpposingPositionOrOrders** is FALSE (No), and the current Trade Position Quantity is less than 0 or the current Trade Position combined with the quantities of all working orders is less than 0, then the Order Action will not be allowed. However, if [SupportReversals](#) is set to TRUE (Yes), then none of this applies.
4. If the Order Action would result in a Trade Position Quantity greater than the [MaximumPositionAllowed](#), then the Order Action will not be allowed. There are two ways this is determined.

If [CancelAllOrdersOnEntriesAndReversals](#) is set to FALSE, then the new order quantity is combined with the current Trade Position Quantity and combined with the quantities of existing working orders for the Symbol and Trade Account to determine if the [MaximumPositionAllowed](#) will be exceeded. If [MaximumPositionAllowed](#) will be exceeded, the Order Action is not allowed.

If [CancelAllOrdersOnEntriesAndReversals](#) is set to TRUE, then the new order quantity is combined with the current Trade Position Quantity and combined with the quantities of existing market orders for the Symbol and Trade Account to determine if the [MaximumPositionAllowed](#) will be exceeded. If [MaximumPositionAllowed](#) will be exceeded, the Order Action is not allowed.

If one of these exceptions is met and the Order Action is not allowed, then the reason for the order not being allowed is displayed in the **Trade >> Trade Service Log**.

Buy Exit

When this Order Action is triggered, a Sell order will be sent to your connected trading service or to the Trade Simulation System, if certain exceptions are not met. The following are the exceptions:

1. If [CancelAllWorkingOrdersOnExit](#) is FALSE (No) and the Trade Position quantity combined with the quantities of Sell working orders are less than or equal to 0, the Order Action will be ignored (meaning there is no Position or there is a Short Position). In this case Buy working orders are ignored in the calculation.
2. If the Trade Position quantity is less than or equal to 0, the Order Action will be ignored.

If one of these exceptions is met and the Order Action is not allowed, then the reason for the order not being allowed is displayed in the **Trade >> Trade Service Log**.

If you are using Attached Orders that will flatten the Trade Position, then it is not possible to use BuyExit unless [CancelAllWorkingOrdersOnExit](#) is set to TRUE (Yes).

If [CancelAllWorkingOrdersOnExit](#) is set to TRUE (Yes), then all working orders for the Symbol and Trade Account the automated trading system is for, will be canceled when a Buy Exit Order Action is performed.

The Trade Position will be flattened by this Order Action unless an order quantity is specified.

In the case of the ACSIL **sc.BuyExit()** function, the order quantity is specified with the **OrderQuantity** member of the [s_SCNewOrder](#) structure.

In the case of the **Spreadsheet System for Trading** study, the quantity can optionally be specified with the **Order Quantity** cell (J26). If **Order Quantity** is blank or 0, then the current Trade Position will be flattened.

If an order quantity is specified and it exceeds the current Trade Position Quantity minus any working exit orders that already exist, the order quantity will be reduced to match the current Trade Position Quantity minus any working exit orders that already exist. For example, if the Trade Position Quantity is +2 and there is one working Sell order with a quantity of 1, and the specified order quantity from the ACSIL or Spreadsheet trading system is 3, the new order quantity will be reduced to 1.

A **BuyExit** does not need to be used. A Trade Position can be manually exited or you can use the [Support Attached Orders For Trading](#) variable in the case of an ACSIL trading study. In the case of the **Spreadsheet System for Trading** study, refer to the section [Initiating Trades with Auto-Trading and Using Manual or Attached Orders Exits](#).

Using Exits Only for a Specialized Exit Strategy: You may want to develop a trading system for exiting manually entered Positions that only use the **BuyExit** and [SellExit](#) Order Actions. For example, you may manually enter a Position and then you want to have an exit based upon a formula involving a particular study Subgraph. In this case you would not use the [BuyEntry](#) or [SellEntry](#) Order Actions. You would only use a formula that would trigger a **BuyExit** or [SellExit](#). This provides very advanced exit trade management.

Sell Entry

When the Sell Entry Order Action is triggered, a Sell order will be sent to your connected trading service or to the Trade Simulation System, with the specified quantity as long as certain exceptions are not met.

These exceptions do not apply when using [Unmanaged Automated Trading](#).

The following are the exceptions when using managed automated trading (the default):

1. When [AllowMultipleEntriesInSameDirection](#) is set to FALSE (No), and the current Trade Position Quantity is less than 0, then the Order Action will not be allowed.

Or when [AllowMultipleEntriesInSameDirection](#) is set to FALSE (No), [CancelAllOrdersOnEntriesAndReversals](#) is set to FALSE (No), and the current Trade Position Quantity combined with the quantities of all working orders is less than 0, then the Order Action will not be allowed.

Or when [AllowMultipleEntriesInSameDirection](#) is set to FALSE (No), [CancelAllOrdersOnEntriesAndReversals](#) is set to TRUE (Yes) and the Current Trade Position Quantity combined with the quantity of working market orders is less than 0, then the Order Action will not be allowed.

Note: When the quantities of working orders are added together and combined with the Trade Position Quantity there are some special rules in the calculations to be aware of. Pending Child orders are ignored. In the case of Target and Stop orders, which are orders in an OCO Group, only the quantity of one of these is used, since one will cancel the other.

2. If [AllowEntryWithWorkingOrders](#) is set to FALSE (No) and there are working orders existing for the symbol, then the Order Action will not be allowed.
3. If **AllowOppositeEntryWithOpposingPositionOrOrders** is FALSE (No), and the current Trade Position Quantity is greater than 0 or the current Trade Position Quantity combined with the quantities of all working orders is greater than 0, then the Order Action will not be allowed. However, if [SupportReversals](#) is set to TRUE (Yes), then none of this applies.
4. If the Order Action would result in a Trade Position Quantity more negative than [MaximumPositionAllowed](#) * -1, then the Order Action will not be allowed. There are two ways this is determined.

If [CancelAllOrdersOnEntriesAndReversals](#) is set to FALSE, then the new order quantity is combined with the current Trade Position Quantity and combined with the quantities of existing working orders for the Symbol and Trade Account to determine if the [MaximumPositionAllowed](#) will be exceeded. If [MaximumPositionAllowed](#) will be exceeded, the Order Action is not allowed.

If [CancelAllOrdersOnEntriesAndReversals](#) is set to TRUE, then the new order quantity is combined with the current Trade Position Quantity and combined with the quantities of existing market orders for the Symbol and Trade Account to determine if the [MaximumPositionAllowed](#) will be exceeded. If [MaximumPositionAllowed](#) will be exceeded, the Order Action is not allowed.

If one of these exceptions is met and the Order Action is not allowed, then the reason for the order not being allowed is displayed in the **Trade >> Trade Service Log**.

Sell Exit

When this Order Action is triggered, a Buy order will be sent to your connected trading service or to the Trade Simulation System, if certain exceptions are not met. The following are the exceptions:

1. If [CancelAllWorkingOrdersOnExit](#) is FALSE (No) and the Trade Position quantity combined with the quantities of Buy working orders are greater than or equal to 0, the Order Action will be ignored (meaning there is no Position or there is a Long Position). In this case Sell working orders are ignored in the calculation.
2. If the Trade Position quantity is greater than or equal to 0, the Order Action will be ignored.

If one of these exceptions is met and the Order Action is not allowed, then the reason for the order not being allowed is displayed in the **Trade >> Trade Service Log**.

If you are using Attached Orders that will flatten the Trade Position, then it is not possible to use SellExit unless [CancelAllWorkingOrdersOnExit](#) is set to TRUE (Yes).

If [CancelAllWorkingOrdersOnExit](#) is set to TRUE (Yes), then all working orders for the Symbol and Trade Account the automated trading system is for, will be canceled when a Sell Exit Order Action is performed.

The Trade Position will be flattened by this Order Action unless an order quantity is specified.

In the case of the ACSIL **sc.SellExit()** function, the order quantity is specified with the **OrderQuantity** member of the [s_SCNewOrder](#) structure.

In the case of the **Spreadsheet System for Trading** study, the quantity can optionally be specified with the **Order Quantity** cell (J26). If **Order Quantity** is blank or 0, then the current Trade Position will be flattened.

If an order quantity is specified and it exceeds the current Trade Position Quantity minus any working exit orders that already exist, the order quantity will be reduced to match the current Trade Position Quantity minus any working exit orders that already exist. For example, if the Trade Position Quantity is -2 and there is one working Buy order with a quantity of 1, and the specified order quantity from the ACSIL or Spreadsheet trading system is 3, the new order quantity will be reduced to 1.

A **SellExit** does not need to be used. A Trade Position can be manually exited or you can use the

[Support Attached Orders For Trading](#) variable in the case of an ACSIL trading study. In the case of the **Spreadsheet System for Trading** study, refer to the section [Initiating Trades with Auto-Trading and Using Manual or Attached Orders Exits](#).

Using Exits Only for a Specialized Exit Strategy: You may want to develop a trading system for exiting manually entered Positions that only use the [BuyExit](#) and **SellExit** Order Actions. For example, you may manually enter a Position and then you want to have an exit based upon a formula involving a particular study Subgraph. In this case you would not use the [BuyEntry](#) or [SellEntry](#) Order Actions. You would only use a formula that would trigger a [BuyExit](#) or **SellExit**. This provides very advanced exit trade management.

Using Multiple Automated Trading Systems at the Same Time

Using multiple automated trading systems at the same time within a single copy of Sierra Chart is supported, as long as the trading systems are for different Symbols or for different Trade Accounts if using the same symbol.

The Trade Account that an automated trading system uses is based upon the Trade Account selected on the Trade Window for the chart the trading system is applied to. Refer to [Selecting Trade Account](#) for instructions to change the Trade Account the automated trading system will use.

When using 2 or more automated trading systems for the same Symbol, each needs to be using a different Trade Account.

If the trading systems are for the same Symbol and same Trade Account, then there is potential for conflict. When the **Automated Trading Management** logic as described on this page calculates the quantities of Working orders for the Symbol and Trade Account, and uses the Trade Position Quantity for the symbol and Trade Account, it is for the Symbol and Trade Account independent of a specific trading system you are running. These quantity calculations are not specific to a particular chart or trading system. They are only specific for a particular Symbol and Trade Account.

Therefore, the two or more trading systems can potentially conflict with each other unless you are able to understand and resolve these conflicts. It is beyond the scope of this document and our support to provide any assistance with this. It is highly recommended not to use 2 or more automated trading systems for the same Symbol and Trade Account at the same time, unless you can understand and resolve the potential conflicts.

Differences Between Live Automated Trading and Simulated Automated Trading

Your automated trading system can perform differently with Live trading compared to Simulated trading. Here are some reasons why:

- Live orders do not fill instantly as they do in Trade Simulation Mode assuming the order can be filled instantly in Trade Simulation Mode based upon its [Order Type](#), its price and the current Bid, Ask, Last values. With live trading an order will take time to fill even though it may happen very quickly. You need to consider this delay and how your

trading system will be affected by it.

For example, until an order fills, the Trade Position Quantity will not change. It is not possible to check the Position Quantity immediately after submitting an order and expect a change because this will not happen.

- When an order does fill, there is additional time required to get a notification of the fill with live trading compared to Trade Simulation Mode. You need to consider this delay and how your trading system will be affected by it.

While this is very unlikely, there is also the possibility that the notification of the fill is not received. In which case the Trade Position Quantity might not be accurate for the symbol. Or it could be temporarily inaccurate.

If for some reason the Trade Position Quantity is not reflecting the correct actual Position Quantity, Sierra Chart will synchronize it to the Trade Position data from your Trading service when your trading service automatically updates the Trade Position data.

- An order that is submitted, could be rejected and will never fill. This is something very important to understand. There can be various reasons why an order gets rejected during live trading.
- An order can fill partially. And there can be many partial fills. The entire quantity may not fill all at once.
- With live/non-simulated trading, when an order is canceled there will be a delay, for the Status of the order to become canceled. When an order is modified, there will be a delay, for the price or quantity to change for the order. And there is also delay for the Status to go back to Open for the order.

In the case of Trade Simulation Mode, order cancellations and modifications are performed immediately within Sierra Chart since the cancellation or modification does not have to be sent to an external service.

- Orders can have many order statuses. The status of an order is updated by the trading service and status updates do not happen instantly and there can be some delay. While this is unlikely, there is also the potential that order status could be incorrect.
- There are differences in the timing of each chart update/calculation during real-time chart updating as compared to chart updating during a back test or replay. For more information, refer to [Differences Between Back Testing and Real-Time Auto Trade System Evaluation](#).
- While timing differences may seem small and insignificant, depending upon how your auto trade system is designed, it can make a significant difference.

You should also understand that using a trade simulation environment provided by your Trading service will never duplicate the live market. You can only truly understand how your trading system will perform in a live environment by actually testing it. You should trade just a single contract or share, and watch it very carefully.

Automatic Recovery of Trading System After Restart

This section explains the steps you should take to set up Sierra Chart when using an automated trading system to have it automatically continue to run after Sierra Chart is restarted.

Related to this, when Sierra Chart loses the connection to the connected Trading service, it will continuously try to reconnect if

Global Settings >> Data/Trade Service Settings >> Reconnect on Failure is enabled. When Sierra Chart can reconnect, your automated trading system will continue. It will not be disabled.

1. Optional Step: Set up a scheduled task in Windows to start Sierra Chart. It should be started every minute. When Sierra Chart is started by the scheduled task and it is already running, another instance will not start.
2. Select **Global Settings >> Data/Trade Service Settings**. Enable the **Connect On Program Startup** and **Reconnect on Failure** options.
3. When Sierra Chart automatically connects to the specified Trading service on program startup, after that connection succeeds, all of the Open orders and Trade Positions will be requested from the remote server after 5 seconds. Previous to this 5 seconds, the previously remembered Open orders and Trade Positions are available to the automated trading system to rely on.
4. Save the chart that contains the automated trading system as part of a Chartbook. Refer to the [Saving Chartbook](#) documentation.
5. Add the Chartbook to automatically open through **Global Settings >> General Settings >> Files to Open on Startup**.
6. Make sure there is no checkmark by **Trade >> Disable Auto Trading on Startup**.
7. Make sure there is no checkmark by **Trade >> Trade Simulation Mode On**, if you want orders submitted to the connected Trading service.
8. Make sure there is no checkmark by **Trade >> Trading Locked**, if you want orders submitted to the connected Trading service.
9. In the case of an ACSIL trading system, you may want to use the [sc.StorageBlock member](#) of the ACSIL structure to store persistent data.
10. You may want to set your Chartbook to automatically save periodically through **Global Settings >> General Settings >> Charts >> Chartbooks >> Autosave Chartbooks Inter**

Troubleshooting Automated Trading System Behavior

When your automated trading system is not functioning as you expect, you need to follow a troubleshooting procedure appropriate for what you are doing.

The information in this section is also generally applicable to general custom study development, and alert formula programming.

First understand that Sierra Chart support does ***not*** provide automated trading system help, development, debugging services, or debugging of complex study configurations, even for a cost. This is not part of our offerings, whether for free or for a payment, and it is not possible for us to do this for various reasons as explained.

- We did not create the automated trading system or know any of its code, design,

formulas or logic. Only you understand your automated trading system and what it does. When you have slowly built up your automated trading system over time, you have a thorough understanding of it every step of the way.

If it involves a complex study configuration, you have an understanding of each element of the configuration and have built that up incrementally over time. When we are presented with a complex configuration and/or program code suddenly at once, it is immensely complicated to us and we will not understand how it is all configured without an extensive time allocation on our part which is not at all part of our offerings.

- We do not know what the final result is you are trying to accomplish. Even if this is expressed to us, all the other points in this section are still relevant, and still there are a lot of fine details, that you know in your mind that we do not.
- We do not know how the Automated Trading Management [Variables](#) are set for your automated trading system.
- In the case of when using the **Spreadsheet System for Trading** study we do not know the other Input settings you are using. In the case of ACSIL, we do not know other ACSIL variable settings you may be using.
- We do not know what other studies are on the chart, how they are configured and what you expect from them.
- We cannot visually see the behavior of your automated trading system during a replay or during real-time updating. In addition to this, we do not know the price and volume actions and values during the replay or during real-time updating.
- We cannot see the price data in the chart, the studies on the chart, and the results of those studies.
- In the case of when data in other charts is being referenced by an ACSIL study or you when using the **Study/Price Overlay** study in the chart, we do not know whether those referenced charts are open, what data they contain, and what data is being referenced in them, and the Chart Settings for those charts. All of this needs to be understood and understood well to understand why there is a problem that involves the data being referenced in the other charts.
- In the case of a Spreadsheet, we cannot see the input/referenced values of a formula, and be able to view the [Formula Details](#).
- We do not know the order actions that your automated trading system is performing or has performed. We do not know the context of these order actions.

For example, if a particular order action was not even performed like a Buy Entry, then only you will know the answer why. There is no way we can possibly know that. If it is an ACSIL based trading system, you need to add debugging information through the use of the [sc.AddMessageToLog](#) function to analyze your trading system.

- In the case of an ACSIL trading system, we do not know the return values of the of the order action functions at the time they are called (Example: **sc.BuyEntry**). You will need to add debugging code to analyze these. Refer to [Debugging ACSIL Trading Systems](#).
- We cannot see the **Trade >> Trade Activity Log >> Trade Activity** tab, **Trade >> Trade Service Log**, or the **Window >> Alerts Manager >> Alerts Log**. All of which are critical to understand in part what an automated trading system is doing

and has done. The [Trade >> Trade Activity Log](#) is the most important of these.

- There are [Differences Between Live Automated Trading and Simulated Automated Trading](#), and there is debugging required to understand how those differences are affecting the trading system. This is your responsibility to analyze.
- When performing non-simulated trading and your trading system is sending orders to and receiving updates from an external trading service, then there are numerous potential problems, trading account limitations, and behaviors with that external service and network connectivity which can lead to unexpected behavior. For example, an order your automated trading system sent was rejected by the trading service or did not get a response.
- We are not being paid for this level of help and we do not have the time to provide this level of help. And if we did take the time to help, this would take away substantial time from the development of Sierra Chart itself which benefits the entire user base.

So when we are asked about a problem with an automated trading system you can understand that we are at a major disadvantage. For us to even begin to answer some of the questions we are asked, requires a large amount of information, understanding, and a major time commitment, to even give an answer. We would have to gain all of the knowledge of what you are doing from where you started. We would have to be able to interact with the automated trading system and studies and visually see what it is doing.

All of this is immensely far outside the scope of our support especially with the low prices of Sierra Chart. Therefore, in almost all cases we cannot help other than refer you to the relevant documentation. And hypothetically even if we did get involved in this, you would be charged at a rate of 80-150 USD per hour and we would need to have an interest in getting involved in this which we definitely do not unless there is a technical problem which needs to be resolved.

Once again, we do not have the time or the interest to be involved in this. Our main interest is to provide software which meets the general requirements of our users, have it be stable, capable and well-documented. So from this perspective, any question you have for us should be made with this in mind or not made at all.

In general, with the type of questions we see that fall within the category of this subsection, our answer to you is you tell us what the problem is. The basis of that answer is we do not know what the issue is. We do not have the level of understanding that you do about what you have created and a significant part of the functioning is based upon what you have created and configured.

If you have a very specific question for Sierra Chart support and you provide clear surrounding context information, then in some cases Sierra Chart support can help with that assuming the issue is greatly simplified and presented in a simple and understandable way.

You need to watch what your trading system is doing in real time and also during a slow speed [Replay Back Test](#) to understand what it does and why.

When the trading system takes some action these actions are logged in detail in the [Trade >> Trade Activity Log](#), unless the order action was ignored in which case the reason for this will be given in the **Trade >> Trade Service Log**. The **Trade Activity Log** is one of the most useful logs to understand what your automated trading system is doing.

If an automated order action is ignored, the reason for this will be given in the **Trade >> Trade Service Log**. Refer to this log for the reason why. For additional information, refer to [Trade Service Log](#).

When using the **Spreadsheet System for Trading** study, refer to [Ignored Signals With Spreadsheet Systems or Alerts](#) for further information.

It may also be helpful to go through the [Spreadsheet System for Trading Test Procedure](#) to help with understanding how a basic Spreadsheet automated trading system will work and indirectly provide help to troubleshoot your automated Spreadsheet trading system.

For an ACSIL automated trading system, refer to [Debugging/Troubleshooting Automated Trading Systems](#).

When using the **Trading System Based on Alert Condition** study, it is recommended to follow through the instructions for the [Trading System Based on Alert Condition Study](#).

One potential issue that can lead to unexpected and unusual behavior, is not having the correct calculation order of studies. For complete details about this subject and what to do, refer to [Study Calculation Precedence And Related Issues](#).

Now hopefully all of the above information will help you to debug/troubleshoot your automated trading system or your study configuration.

Issues and Improvements in Sierra Chart related to Automated Trading

Regarding all of the above information, we understand, that there could be a potential issue in Sierra Chart or improvement relating to automated trading or studies which needs to be addressed by us.

In order for us to become aware of this, it is necessary that this is communicated to us by a user who is competent and presents the information in a way in which we can understand. It needs to be very clear that there is some issue or improvement that needs to be worked on by us.

The information must be presented in a simple, clear and orderly way. And without over complicating it and providing an excess amount of information. And once again this must come from someone who is competent and can present the issue properly, simply, and effectively. Otherwise, it is just going to get disregarded as a programming issue on the user side or a misunderstanding, which is the case almost always.

Often when we are presented with an automated trading system problem that a user is encountering, additional documentation related to it is written and improvements are made to logging of automated trading related messages/entries in the **Trade Service Log** and [Trade Activity Log](#). Therefore in these cases where it has been determined that there is no issue to solve in Sierra Chart or an improvement needed, instead clarifications were made in order to make it clear to the user what the source of the problem is.

Finally, if after reading all of the above, you believe Sierra Chart support still needs to help you with your automated trading system, then do not use Sierra Chart to create an automated trading system because

clearly it is beyond your abilities. It would be best that you find a competent developer to help you with this instead.

Understanding Calculation of Trading System Studies

An automated trading system is programmed as a study using either the [Spreadsheet System for Trading](#) study or the [ACSIL Trading Functions](#) in an Advanced Custom Study that you write.

When a chart is updated with data, during real-time updating, during a replay or during a back test, the chart bars are updated and other market data structures, and after that the studies on the chart are calculated. It is at that time, that the automated trading study can then do its calculations and perform trading related actions.

If you want to perform some trading related action immediately when a bar closes or at some other point in time, then in the case of an Advanced Custom Study you may want to set [sc.UpdateAlways](#) to TRUE to ensure that the study function is calculated at every [Chart Update Interval](#) so it does not have to wait until new market data is received.

Low Latency Automated Trading

The first step to reduce the amount of time from when the market data is received to the time an order is submitted, is to reduce the [Chart Update Interval](#) for the chart which contains the automated trading study. This can be set as low as 10 milliseconds.

If you require lower latency automated trading where you need to immediately submit orders in response to changing market data, then it is recommended to use the [DTC protocol Server](#) and develop an independent automated trading program which uses the DTC server for market data and trading functionality. You can use any programming language that you require in this particular case.

Order of Execution

The order of execution of studies and automated trading system studies is according to the rules all documented in the [Study Calculation Precedence](#) section.

When using the **Spreadsheet System for Trading** study, the order of execution of the different Formula columns is documented in the [Order of Formula Column Processing and Internal Calculation Order](#) section.

Orders have a [Status](#) field. The order could be in a pending state or in an open state or in another state. Based upon the status of the order, a particular action on the client-side may be delayed and cannot be immediately performed. For example, until an order goes from an **Order Sent** to an **Open** state, it cannot be modified. This is only one example. Although Attached Orders have different behavior.

Any further questions about this you need to refer to the relevant [documentation](#) of the functionality you are working with.

