

Using Drawing Tools From an Advanced Custom Study

- [Introduction](#)
- [Using Tools with sc.UseTool\(\)](#)
 - [Preventing More Than One Chart Drawing Per Chart Bar or More Drawings than Intended](#)
 - [Return Value](#)
 - [Code Example](#)
 - [Adjusting Existing Chart Drawings](#)
- [Drawing Tools and s_UseTool Member Descriptions](#)
 - [s_UseTool::DrawingType](#)
 - [s_UseTool::AddMethod](#)
 - [s_UseTool::ChartNumber](#)
 - [s_UseTool::LineNumber](#)
 - [s_UseTool::BeginDateTime](#)
 - [s_UseTool::EndDateTime](#)
 - [s_UseTool::ThirdDateTime](#)
 - [s_UseTool::BeginIndex](#)
 - [s_UseTool::EndIndex](#)
 - [s_UseTool::ThirdIndex](#)
 - [s_UseTool::UseRelativeVerticalValues / s_UseTool::UseRelativeValue](#)
 - [s_UseTool::BeginValue](#)
 - [s_UseTool::EndValue](#)
 - [s_UseTool::ThirdValue](#)
 - [s_UseTool::Region](#)
 - [s_UseTool::Color](#)
 - [s_UseTool::SecondaryColor](#)
 - [s_UseTool::TransparencyLevel](#)
 - [s_UseTool::SquareEdge](#)
 - [s_UseTool::LineWidth](#)
 - [s_UseTool::LineStyle](#)
 - [s_UseTool::Text](#)
 - [s_UseTool::DisplayHorizontalLineValue](#)
 - [s_UseTool::FontSize](#)
 - [s_UseTool::FontBackColor](#)
 - [s_UseTool::FontFace](#)
 - [s_UseTool::FontBold](#)
 - [s_UseTool::FontItalic](#)
 - [s_UseTool::TextAlignment](#)
 - [s_UseTool::ReverseTextColor](#)

- [s UseTool::MarkerType](#)
- [s UseTool::MarkerSize](#)
- [s UseTool::UseBarSpacingForMarkerSize](#)
- [s UseTool::ShowVolume](#)
- [s UseTool::ShowAskBidDiffVolume](#)
- [s UseTool::ShowPriceDifference](#)
- [s UseTool::ShowTickDifference](#)
- [s UseTool::ShowCurrencyValue](#)
- [s UseTool::ShowPercentChange](#)
- [s UseTool::ShowTimeDifference](#)
- [s UseTool::ShowNumberOfBars](#)
- [s UseTool::ShowAngle](#)
- [s UseTool::ShowEndPointPrice](#)
- [s UseTool::ShowEndPointDateTime](#)
- [s UseTool::ShowEndPointDate](#)
- [s UseTool::ShowEndPointTime](#)
- [s UseTool::MultiLineLabel](#)
- [s UseTool::ShowPercent](#)
- [s UseTool::ShowPrice](#)
- [s UseTool::RoundToTickSize](#)
- [s UseTool::ExtendLeft](#)
- [s UseTool::ExtendRight](#)
- [s UseTool::TimeExpVerticals](#)
- [s UseTool::TimeExpTopLabel1](#)
- [s UseTool::TimeExpBottomLabel1](#)
- [s UseTool::TimeExpBasedOnTime](#)
- [s UseTool::TransparentLabelBackground](#)
- [s UseTool::FixedSizeArrowHead](#)
- [s UseTool::RetracementLevels\[\]](#)
- [s UseTool::LevelColor\[\]](#)
- [s UseTool::LevelWidth\[\]](#)
- [s UseTool::LevelStyle\[\]](#)
- [s UseTool::AddAsUserDrawnDrawing](#)
- [s UseTool::DrawUnderneathMainGraph](#)
- [s UseTool::UseToolConfigNum](#)
- [s UseTool::FlatEdge](#)
- [s UseTool::DrawWithinRegion](#)
- [s UseTool::CenterPriceLabels](#)
- [s UseTool::NoVerticalOutline](#)
- [s UseTool::AllowSaveToChartbook](#)
- [s UseTool::ShowBeginMark](#)
- [s UseTool::ShowEndMark](#)
- [s UseTool::AssociatedStudyID](#)
- [s UseTool::HideDrawing](#)
- [s UseTool::LockDrawing](#)
- [s UseTool::DrawOutlineOnly](#)

- [s_UseTool::NumCycles](#)
- [s_UseTool::ExtendMultiplier](#)
- [s_UseTool::DrawMidline](#)
- [s_UseTool::AllowCopyToOtherCharts](#)
- [sc.ChartDrawingExists\(\)](#)
- [sc.UserDrawnChartDrawingExists\(\)](#)
- [sc.GetUserDrawnChartDrawing\(\)](#)
- [sc.GetLineNumberOfSelectedUserDrawnDrawing\(\)](#)
- [sc.GetUserDrawnDrawingByLineNumber\(\)](#)
- [sc.GetACSDrawingByLineNumber\(\)](#)
- [sc.GetACSDrawingByIndex\(\)](#)
- [sc.GetACSDrawingsCount\(\)](#)
- [sc.DeleteACSCartDrawing\(\)](#)
- [sc.DeleteUserDrawnACSDrawing\(\)](#)
- [RGB Color Values](#)
- [Drawing Chart Drawings On Top of or Underneath Main Graph and Studies](#)
- [Adding Chart Drawings to Other Charts from an ACSIL Study Function](#)
- [Drawing Lines with a Specific Angle Using ACSIL](#)

Introduction

This page documents programmatically adding Chart Drawings and interacting with user drawn Chart Drawings through the [Advanced Custom Study Interface and Language](#).

For information about Drawing Tools, refer to [Chart Drawing Tools](#).

Using Tools with **sc.UseTool()**

Type: ACSIL Function

Declaration: `int UseTool(s_UseTool& UseTool);`

This page documents programmatically adding the same type of Chart Drawings to a chart from Advanced Custom Studies, that are normally drawn by a user using the [Chart Drawing Tools](#). Therefore, Advanced Custom Studies can independently add the same types of Chart Drawings to a chart programmatically.

Refer to the Sierra Chart [Advanced Custom Study Interface and Language](#) (ACSL) page for more information about the Advanced Custom Study Interface and Language.

To use the built-in Sierra Chart Drawing Tools from an Advanced Custom Study, you will use the **sc.UseTool()** function.

When calling this function, a Chart Drawing is added to an internal list in a chart. The **sc.UseTool()** function supports many different drawing types. This is a very powerful feature. You can place various types of Chart Drawing objects on a chart or place Text anywhere on a chart, and dynamically move

those objects. You can control the foreground and background colors and font size of Text.

The **sc.UseTool()** function uses the **s_UseTool** structure to specify the various parameters for the tool. You need to define an instance of the **s_UseTool** structure in your function. **sc.UseTool()** takes a reference to the **s_UseTool** instance you defined in the study function.

Set the relevant members of the **s_UseTool** structure, call the **sc.UseTool()** function and pass an instance of **s_UseTool** to that function. The **DrawingType** member of the **s_UseTool** structure specifies which drawing tool to use.

Before setting the members of **s_UseTool**, always be sure to use the **Clear()** function of the **s_UseTool** structure to initialize the structure. This is especially important if you are using the same instance of the structure to multiple calls of **sc.UseTool()**. Refer to the code example below.

The [s_UseTool Member Descriptions](#) section documents the members of the **s_UseTool** structure for all of the imported drawing tools.

For actual code examples, refer to the **scsf_UseToolExample*** study functions located in the **/ACS_Source/studies.cpp** file in the folder where Sierra Chart is installed to on your system.

For another example to place text on a chart, refer to the **scsf_CountDownTimer** in the **/ACS_Source/studies2.cpp** file.

This paragraph applies when **s_UseTool::AddAsUserDrawnDrawing** is not set which is almost always the case. Chart Drawings added with **sc.UseTool()**, are automatically deleted from the chart when the study is removed from the chart, or when the study is fully recalculated, unless you have set **s_UseTool::AddAsUserDrawnDrawing** to 1 (TRUE). Therefore, when **s_UseTool::AddAsUserDrawnDrawing** is set to 0, there is no need to delete Chart Drawings manually by using **sc.DeleteACSChartDrawing()**.

Expressly deleting a chart drawing with [sc.DeleteACSChartDrawing\(\)](#) is only needed for more specialized purposes.

On a recalculation of a study that has added non-user drawn Chart Drawings through ACSIL will have those Chart Drawings automatically deleted.

Chart drawings added by an ACSIL function are never saved to a Chartbook. They exist only in temporary memory.

Preventing More Than One Chart Drawing Per Chart Bar or More Drawings than Intended

When adding Chart Drawings with **sc.UseTool()** it is important not to add more than one Chart Drawing per chart bar or more than intended unless that is the intention. During a full recalculation of the study, if one Chart Drawing is added per chart bar, there will not be a problem in this case.

However, during normal chart updating [sc.Index](#) is set to the index of the last bar in the chart from the prior chart update. Therefore, it is important not to add another Chart Drawing to the bar at that index if that is not the intention. Even if it is the intention to add more than one Chart

Drawing per bar index, you need to be careful that you do not add more Chart Drawings per chart bar index than intended.

When updating the existing Chart Drawing at a particular bar index or at `sc.Index`, set [s_UseTool::AddMethod](#) to **UTAM_ADD_OR_ADJUST** and specify the same [s_UseTool::LineNumber](#) returned by the **sc.UseTool** function when the Chart Drawing was originally added. You will need to hold this LineNumber in a [Persistent Variable](#).

If you do not set [s_UseTool::AddMethod](#) to **UTAM_ADD_OR_ADJUST** and specify the same [s_UseTool::LineNumber](#) originally returned for a Chart Drawing that you are updating and do not want to add again, then there will be a large number of Chart Drawings added to the chart and this will have a serious negative performance and memory impact. Make sure you are not creating this problem in your code!

Return Value

sc.UseTool() returns 1 on success, 0 on failure.

Code Example

```
// Marker example
s_UseTool Tool;
int UniqueLineNumber = 74191;//any random number.

Tool.Clear(); // Reset tool structure. Good practice but unnecessary in this case.
Tool.ChartNumber = sc.ChartNumber;

Tool.DrawingType = DRAWING_MARKER;
Tool.LineNumber = UniqueLineNumber + 1;

BarIndex = max(0, sc.ArraySize - 35);

Tool.BeginDateTime = sc.BaseDateTimeln[BarIndex];
Tool.BeginValue = sc.High[BarIndex];

Tool.Color = RGB(0,200,200);
Tool.AddMethod = UTAM_ADD_OR_ADJUST;

Tool.MarkerType = MARKER_X;
Tool.MarkerSize = 8;

Tool.LineWidth = 5;

sc.UseTool(Tool);
```

Adjusting Existing Chart Drawings

When a Chart Drawing is added to the chart through the use of the **sc.UseTool** function, it can later be adjusted/modified. The method by which a chart drawing can be adjusted is by calling the **sc.UseTool** with [s_UseTool::AddMethod](#) set to **UTAM_ADD_OR_ADJUST**, which is the default.

When adjusting a drawing, set the **s_UseTool::LineNumber** member to the same number that was previously set after the **sc.UseTool** function returned when the Chart Drawing was originally added. The **LineNumber** that was automatically set can be remembered into a [persistent](#)

[variable](#).

If more than one drawing uses the same LineNumber, only the first found drawing will be adjusted when adjusting a Chart Drawing, for performance reasons. Therefore, it is not considered best practice to use the same **LineNumber** for different Chart Drawings added by the **sc.UseTool** function.

Set the other members of the **s_UseTool** structure that you want to modify. The structure members that you do not want to change need to be left unset. For more information, refer to [AddMethod](#).

It is possible to modify a Chart Drawing that was added as a user drawn drawing. A drawing is considered a user drawn drawing if it was added with **sc.UseTool** and the **s_UseTool::AddAsUserDrawnDrawing** variable is set to 1.

When modifying a user drawn drawing, it is necessary to set **s_UseTool::AddAsUserDrawnDrawing** to modify it.

sc.UseTool returns 1 upon a successful adjustment/modification of an existing chart drawing.

Drawing Tools and s_UseTool Member Descriptions

s_UseTool::DrawingType

Type: DrawingTypeEnum

This member needs to be set to one of the following drawing types, unless you are adjusting an existing drawing:

- **DRAWING_LINE**: Draws a Line on the chart.
- **DRAWING_RAY**: Draws a Ray on the chart.
- **DRAWING_EXTENDED_LINE**: Draws an Extended Line (extends in both directions) on the chart.
- **DRAWING_RECTANGLEHIGHLIGHT**: Draws a rectangle highlight drawing.
- **DRAWING_RECTANGLE_EXT_HIGHLIGHT**: Draws a rectangle highlight drawing that extends either right or left.
- **DRAWING_ELLIPSEHIGHLIGHT**: Draws an ellipse highlight drawing.
- **DRAWING_TRIANGLE**: Draws a triangle drawing.
- **DRAWING_TEXT**: This tool draws text anywhere on the chart and in any chart region. You can use both absolute and relative positioning. You can specify various font properties. A very good example of how to use this tool and what you can do can be found in the **scsf_WoodiesPanel** function in the **/ACS_Source/studies7.cpp** file. This is in the folder where Sierra Chart is installed to on your computer.
- **DRAWING_STATIONARY_TEXT**: This tool draws text on the chart using realtive positions.
- **DRAWING_RETRACEMENT** : This tool draws a 2-point Retracement

drawing on the chart.

- **DRAWING_EXPANSION** : This tool draws a 2-point Expansion drawing on the chart.
- **DRAWING_PROJECTION** : This tool draws a 3-point Projection drawing on the chart.
- **DRAWING_CALCULATOR**: This tool draws a Chart Calculator line on the chart.
- **DRAWING_HORIZONTALLINE**: This tool draws a Horizontal Line drawing on the chart. This horizontal line extends from the left side of the chart window to the right side of the chart window.
- **DRAWING_HORIZONTAL_RAY**: Draws a Horizontal Ray on the chart. This horizontal line extends from `s_UseTool::BeginDateTime` to the right side of the chart window.
- **DRAWING_HORIZONTAL_LINE_NON_EXTENDED**: This tool draws a Horizontal Line on the chart which does not extend. It begins at `s_UseTool::BeginDateTime` and ends at `s_UseTool::EndDateTime`.
- **DRAWING_VERTICALLINE**: This tool draws a Vertical line drawing on the chart.
- **DRAWING_ARROW**: This tool draws an Arrow drawing on the chart.
- **DRAWING_PITCHFORK**: This tool draws a Pitchfork drawing on the chart.
- **DRAWING_PITCHFORK_SCHIFF**: This tool draws a Schiff Pitchfork drawing on the chart.
- **DRAWING_PITCHFORK_MODIFIED_SCHIFF**: This tool draws a Modified Schiff Pitchfork drawing on the chart.
- **DRAWING_TIME_EXPANSION**: This tool draws a 2-point Time Expansion drawing on the chart.
- **DRAWING_TIME_PROJECTION**: This tool draws a 3-point Time Projection drawing on the chart.
- **DRAWING_PARALLEL_LINES**: This tool draws a Parallel Lines drawing on the chart.
- **DRAWING_PARALLEL_RAYS**: This tool draws a Parallel Rays drawing on the chart.
- **DRAWING_LINEAR_REGRESSION**: This tool draws a Linear Regression drawing on the chart.
- **DRAWING_RAFF_REGRESSION_CHANNEL**: This tool draws a Raff Regression Channel drawing on the chart.
- **DRAWING_MARKER**: This tool draws a Marker drawing on the chart. For a Marker drawing, it is necessary to set the `s_UseTool::MarkerType` to specify the actual type of marker.
- **DRAWING_FAN_FIBONACCI**: This tool draws a Fibonacci Fan drawing on the chart.
- **DRAWING_REWARD_RISK**: This tool draws a Reward Risk drawing on the chart.
- **DRAWING_SWING_MARKER**: This tool draws a Swing Marker drawing on the chart.
- **DRAWING_DATE_MARKER**: This tool draws a Date Marker drawing on the

chart.

- **DRAWING_REWARD_RISK** : This tool draws a Reward Risk drawing on the chart. For an example function which demonstrates its use, refer to the `scsf_UseToolExampleRewardRisk()` function in the `/ACS_Source/Studies.cpp` file. The Reward/Risk drawing is a rather complex drawing to add due to the number of options. The example lays out all of the options and documents what they do. The example should be used to understand how to add a **DRAWING_REWARD_RISK** drawing.

s_UseTool::AddMethod

Type: Integer

Specifies whether the Chart Drawing for the drawing tool is always added to the drawing list, not allowed to be added, or whether instead it will update an existing Chart Drawing.

It is important to understand, a Chart Drawing can be added to one of the following lists in a chart: Advanced Custom Study Drawing List, Advanced Custom Study Fast Drawing List (for drawings which are specified to use bar indexes rather than bar date-times), User Drawn Chart Drawing List.

When a drawing is added to one of these lists, and there is a check ahead of time and to see if it already exist to see if the existing drawing should be updated or if the new drawing should just be ignored based upon the **AdMethod** , that check takes time based upon the size of the corresponding drawing list.

No check takes place to see if a Chart Drawing exists in an existing list, when the [s_UseTool::LineNumber](#) is unset and has its default value of -1 in which case it will be automatically set since it is known that this will be a new Chart Drawing.

AddMethod by default is set to **UTAM_ADD_OR_ADJUST**.

Possible values for AddMethod:

- **UTAM_ADD_ALWAYS**: The Chart Drawing will be added to the chart even if another Chart Drawing with the same **LineNumber** already exists. There is no check if the Chart Drawing already exists if it is a non-user drawn Chart Drawing. Therefore, the insertion is very fast.

However, it is not possible to add more than one user drawn drawing (when `s_UseTool::AddAsUserDrawnDrawing = 1`) with the same **LineNumber**. This will fail and 0 will be returned from the **sc.UseTool** function. In this case, there is a check if the Chart Drawing already exists.

- **UTAM_ADD_ONLY_IF_NEW**: The Chart Drawing will only be added if no other Chart Drawing already on the chart has the same **LineNumber**, if **LineNumber** is set.

If **LineNumber** is not set, then the Chart Drawing will always be added.

- **UTAM_ADD_OR_ADJUST**: If the Chart Drawing already exists on the chart as identified by the **LineNumber** member, then that existing Chart Drawing will be adjusted. Otherwise, a new Chart Drawing will be added.

If [LineNumber](#) is not set, then a new Chart Drawing will be added. Upon return from the **sc.UseTool** function, **LineNumber** will be set to the automatically assigned number.

If the drawing is adjusted, then only the **s_UseTool** structure members specified will be updated in the existing Chart Drawing, the others will be left as is.

If the **DrawingType** member is a different type than specified previously for the same **LineNumber**, then the Chart Drawing type will change to the new specified **DrawingType**.

s_UseTool::ChartNumber

Type: Integer

This member is ignored if **s_UseTool::AddAsUserDrawnDrawing** is 0, which is the default.

When [s_UseTool::AddAsUserDrawnDrawing](#) is set to 1 or a nonzero value, then **ChartNumber** specifies the chart number that the drawing will be drawn on. Use 0 to put the drawing on the same chart that the study is applied to.

There are special considerations when setting **s_UseTool::AddAsUserDrawnDrawing** to 1. Be sure to read the documentation for [s_UseTool::AddAsUserDrawnDrawing](#).

The **ChartNumber** of a chart is the number of the chart and this is displayed after the "#" sign on the top line of the chart. Each chart has a unique number identifying itself within its Chartbook.

If the **ChartNumber** specified is not in the Chartbook or that chart is in the process of loading chart data (not downloading historical data), then the Chart Drawing will not be added by the **sc.UseTool** function.

s_UseTool::LineNumber

Type: Integer

The **LineNumber** is a unique integer identifier used to identify the Chart Drawing added with the **sc.UseTool** function. This **LineNumber** is used to later identify the Chart Drawing.

This **LineNumber** is needed when performing modifications to the Chart Drawing. Or deleting the Chart Drawing if needed in special cases.

Setting the **LineNumber** should only be done when modifying or deleting an existing Chart

Drawing. Otherwise, leave it unset and let it be automatically assigned. It is considered proper practice to let the `LineNumber` be automatically assigned.

If a **LineNumber** is not specified, it will be automatically assigned as a positive number, or usually negative in the case of user drawn drawing, by ACSIL after calling the **sc.UseTool** function. You will be able to get the value by accessing the **LineNumber** member variable, after calling the **sc.UseTool** function.

The default value for **LineNumber** is -1 which is a flag to the `sc.UseTool` function to automatically assign the `LineNumber`. So a -1 is acceptable and is the default if **LineNumber** is not set.

The **LineNumber** for ACSIL added drawings will not conflict with user drawn chart drawings when **s_UseTool::AddAsUserDrawnDrawing** is set to 0.

It is not considered acceptable practice to use the same **LineNumber** for different Chart Drawings added by the **sc.UseTool** function.

In the case of a user drawn drawing (**s_UseTool::AddAsUserDrawnDrawing**=1) and when **s_UseTool::AddMethod** == **UTAM_ADD_ALWAYS**, using the same `LineNumber` for a drawing will result in **sc.UseTool** returning 0 and the drawing will not be added.

In the case of a user drawn drawing (**s_UseTool::AddAsUserDrawnDrawing**=1), adding a new Chart Drawing, and specifying a negative line number other than -1, will result in **sc.UseTool** returning 0 and the drawing will not be added.

After calling the **sc.UseTool** function and if it returns a nonzero number, the **LineNumber** if it was not set, will be set. You can then check what the **LineNumber** is and set it into a [Persistent Variable](#) if needed for future use.

When calling the **sc.UseTool** function and the [AddMethod](#) is set to **UTAM_ADD_OR_ADJUST**, the Chart Drawing with the specified **LineNumber** will be [adjusted](#) instead.

In the case when `LineNumber` is automatically set when **s_UseTool::AddAsUserDrawnDrawing** is set to 1 or a nonzero number, the automatically assigned `LineNumber` may be a negative number or it may be a positive number. When adding Chart Drawings during the calling of the study function, it will continuously decrement to a more negative number, or decrement to a less positive number, depending whether it is negative or positive, without any skips of numbers. Therefore, when adding a range of drawings you can remember the beginning `LineNumber` and the ending `LineNumber` to later be able to reference that range of drawings for modification or deletion. Remember these values in a [Persistent Variable](#).

s_UseTool::BeginDateTime

Type: [SCDateTime](#)

The Date and Time at which the chart drawing begins.

This member is not used with the Horizontal line tool.

To use a value relative to the left side of the chart for BeginDateTime rather than an absolute Date Time value, specify an integer value from 1 to 150. Where 1 represents the left side of the chart window and where 150 represents the right side of the chart window, not including the Values Scale on the right side of the chart.

Drawing Text After the End of the Chart Bars: It is possible to draw text after the end of the chart bars. This only applies in the case of when the **Tool** member of s_UseTool is **DRAWING_TEXT**. To do this use an integer value of -1, -2, -3 or lower. A value of -1 will display the text in the fill space after the very last bar in the chart, and a value of -2 or lower will display the text in the fill space even when the chart is scrolled back towards the left. When drawing text in the fill space on the right side of the chart, by default the text is one bar spacing beyond the last bar. To increase the spacing use a value less than -2 such as -3. -3 will draw the text in the fill space and it will be offset from the last bar by 2 bar spacings.

Drawing Chart Drawings In The Fill Space (Forward Projection Area): If you want the beginning point of your chart drawing to be in the fill space or forward projection area on the right side of the chart after the last loaded bar in the chart, then use code similar to the example below. This only applies to non-text drawings. This does not work with Text drawings.

```
s_UseTool UseTool;  
UseTool.Clear();  
  
// BeginDateTime is at the second column in the forward projection area  
UseTool.BeginDateTime = sc.BaseDateTimeIn[sc.ArraySize + 1];
```

For maximum performance, use the [BeginIndex](#) member instead of **BeginDateTime**.

s_UseTool::EndDateTime

Type: [SCDateTime](#)

The Date and Time at which the chart drawing ends. In the case of an extending chart drawing, this is the date and time which the drawing extends from.

This is not used with the Horizontal Line, Vertical Line, or Text tools.

EndDateTime should not be the same as **BeginDateTime** because it is required that a drawing spans at least more than one bar unless it is just a Line which is vertical.

To use a value relative to the left side of the chart for EndDateTime rather than an absolute Date Time value, specify an integer value from 1 to 150. Where 1 represents the left side of the chart window and where 150 represents the right side of the chart window, not including the Values Scale on the right side of the chart.

Drawing Chart Drawings In The Fill Space (Forward Projection Area): If you want the ending point of your chart drawing to be in the fill space or forward projection area on the right side of the chart after the last loaded bar in the chart, then use code similar to the example below. This only applies to non-text drawings. This does not work with Text drawings.

```
s_UseTool UseTool;  
UseTool.Clear();  
UseTool.EndDateTime = sc.BaseDateTimeIn[sc.ArraySize + 4];
```

For maximum performance, use the [EndIndex](#) member instead of **EndDateTime**.

s_UseTool::ThirdDateTime

Type: [SCDateTime](#)

For drawings with three points, ThirdDateTime specifies the Date and Time of the third point. Valid for **DRAWING_PROJECTION**, **DRAWING_PITCHFORK**, **DRAWING_PITCHFORK_SCHIFF**, **DRAWING_PITCHFORK_MODIFIED_SCHIFF**, **DRAWING_TRIANGLE**, **DRAWING_TIME_PROJECTION**, **DRAWING_PARALLEL_LINES**, and **DRAWING_PARALLEL_RAYS**. Specified in same manner as **EndDateTime**.

s_UseTool::BeginIndex

Type: Integer

Drawings may be specified with a bar Index instead of a Date-Time to specify the horizontal position.

Using this member instead of [BeginDateTime](#) will be dramatically more efficient, and allows for bars with the same timestamp to be distinguished. The efficiency benefit is only when the drawing is not a [user drawn](#) drawing.

BeginIndex is used to specify the first point of a drawing.

When using **BeginIndex** to modify a drawing, make sure to clear s_UseTool::BeginDateTime by calling the Clear() function on that SCDateTime variable, in case it has a set Date-Time value. Example: UseTool.BeginDateTime.Clear().

```
s_UseTool Tool;  
Tool.BeginIndex = sc.Index;
```

s_UseTool::EndIndex

Type: Integer

Drawings may be specified with a bar Index instead of a Date-Time to specify the horizontal position.

Using this member instead of [EndDateTime](#) will be dramatically more efficient, and allows for bars with the same timestamp to be distinguished. The efficiency benefit is only when the drawing is not a [user drawn](#) drawing.

EndIndex is used to specify the second point of a drawing.

EndIndex should not be the same as **BeginIndex** because it is required that a drawing spans at least more than one bar unless it is just a Line which is vertical.

When using **EndIndex** to modify a drawing, make sure to clear s_UseTool::EndTime by calling the Clear() function on that SCDatetime variable, in case it has a set Date-Time value. Example: UseTool.EndDateTime.Clear().

s_UseTool::ThirdIndex

Type: Integer

When using bar indexes to define drawing points, **ThirdIndex** specifies the bar index of the third point for drawings that used three points. Valid for **DRAWING_PROJECTION**, **DRAWING_PITCHFORK**, **DRAWING_PITCHFORK_SCHIFF**, **DRAWING_PITCHFORK_MODIFIED_SCHIFF**, **DRAWING_TRIANGLE**, **DRAWING_TIME_PROJECTION**, **DRAWING_PARALLEL_LINES**, and **DRAWING_PARALLEL_RAYS**.

When using **ThirdIndex** to modify a drawing, make sure to clear s_UseTool::ThirdDateTime by calling the Clear() function on that SCDatetime variable, in case it has a set Date-Time value. Example: UseTool.ThirdDateTime.Clear().

s_UseTool::UseRelativeVerticalValues / s_UseTool::UseRelativeValue

Type: Integer

Set **UseRelativeVerticalValues** to 1 or any nonzero value, to use **BeginValue** and **EndValue** as relative vertical scale values to the Chart Region instead of as absolute scale values.

Refer to the [BeginValue](#) and [EndValue](#) documentation for more details.

It is supported to use **BeginValue** and **EndValue** as relative vertical scale values to the Chart Region by setting **UseRelativeVerticalValues** to 1 and still use [BeginDateTime](#) and [EndDateTime](#) as absolute references to specific Date-Times.

UseRelativeVerticalValues does not apply to any of the *DateTime or *Index members.

When using **UseRelativeVerticalValues**, it is necessary for there to be a scale established

for the particular Chart Region this study is set to display in through the [sc.GraphRegion](#) setting. Otherwise, the Chart Drawing will not display or display properly. This scale can be established by another study or price graph in the Chart Region, or if there is no other study it must be established by the study using **UseRelativeVerticalValues** by filling in values into a visible [sc.Subgraph](#) or by using a [user-defined scale](#).

s_UseTool::BeginValue

Type: Float

The vertical axis chart value for the beginning point of the Chart Drawing. This value needs to be based on the values of the main price graph or the first study graph in the Chart Region (specified with the **Region** member) where the Chart Drawing will be located.

This is not used with the Vertical line tool.

If **UseRelativeVerticalValues** is set to 1 or a nonzero value, then **BeginValue** is a value relative to the Chart Region itself completely independent of the scale used for the graphs in the Chart Region. It is a percentage. Where 1 = 1%. The entire height of the Chart Region is 100%. 0 = The bottom of the Chart Region. 100 = The top of the Chart Region. These values apply to the Chart Region and not the entire height of the chart window unless there is only a single Chart Region displayed.

The horizontal axis equivalent to **UseRelativeVerticalValues** is by setting the [BeginDateTime](#) member to a integer value between 1 to 150.

s_UseTool::EndValue

Type: Float

The vertical axis chart value for the ending point of the Chart Drawing. This is not needed for the Text tool. This value needs to be based on the values of the main price graph or the first study graph in the Chart Region (specified with the **Region** member) where the Chart Drawing will be located.

This is not used with the Horizontal or Vertical line tools.

If **UseRelativeVerticalValues** is set to 1 or a nonzero value, then **EndValue** is a value relative to the Chart Region itself completely independent of the scale used for the graphs in the Chart Region. It is a percentage. Where 1 = 1%. The entire height of the Chart Region is 100%. 0 = The bottom of the Chart Region. 100 = The top of the Chart Region. These values apply to the Chart Region and not the entire height of the chart window unless there is only a single Chart Region displayed.

The horizontal axis equivalent to **UseRelativeVerticalValues** is by setting the [EndDateTime](#) member to a integer value between 1 to 150.

s_UseTool::ThirdValue

Type: Float

For drawings with three points, the **ThirdValue** member specifies the vertical scale value of the third point.

Valid for **DRAWING_PROJECTION**, **DRAWING_PITCHFORK**, **DRAWING_PITCHFORK_SCHIFF**, **DRAWING_PITCHFORK_MODIFIED_SCHIFF**, **DRAWING_TIME_PROJECTION**, **DRAWING_PARALLEL_LINES**, and **DRAWING_PARALLEL_RAYS**.

Specified in same manner as **EndValue**.

s_UseTool::Region

Type: integer

This is the region on the chart where the Chart Drawing appears. This number is 0 based and must be 0 to 12.

Region 0 represents Chart Region 1. This is where the Main Price Graph is located. Regions 1 to 11 correspond to chart regions 2 to 12. This is where the study graphs below the Main Price Graph are located.

s_UseTool::Color

Type: unsigned integer (COLORREF)

The Chart Drawing color in [RGB format](#). The default value of this is RGB(0, 0, 0) which is equal to black. This variable must be set, otherwise you will not see the Chart Drawing on a black background.

For the Rectangle, Ellipse, and Triangle Chart Drawings (**DRAWING_RECTANGLEHIGHLIGHT**, **DRAWING_RECTANGLE_EXT_HIGHLIGHT**, **DRAWING_ELLIPSEHIGHLIGHT**, **DRAWING_TRIANGLE**), this member specifies the outline color of the Chart Drawing. In order to see the outline, it is necessary to set the **LineWidth** to 1 or greater. Otherwise, the outline will not be visible.

s_UseTool::SecondaryColor

Type: unsigned integer (COLORREF)

The fill color in [RGB format](#). This applies to the following Chart Drawing types: **DRAWING_RECTANGLEHIGHLIGHT**, **DRAWING_RECTANGLE_EXT_HIGHLIGHT**, **DRAWING_ELLIPSEHIGHLIGHT**, **DRAWING_TRIANGLE**.

s_UseTool::TransparencyLevel

Type: Integer

TransparencyLevel sets the amount of transparency for the interior fill area for the following Chart Drawing types: **DRAWING_RECTANGLEHIGHLIGHT**, **DRAWING_RECTANGLE_EXT_HIGHLIGHT**, **DRAWING_ELLIPSEHIGHLIGHT**, **DRAWING_TRIANGLE**.

0 means that the interior fill is completely solid and has no transparency. 100 means that the interior fill is completely transparent and not even visible.

s_UseTool::SquareEdge

Type: Integer

Set this to 1 to use square edges for the ends of a Line or Ray.

s_UseTool::LineWidth

Type: Integer

The line width of the Chart Drawing in pixels. The default is 1. However, you should always set this value so the code is clear.

For the Rectangle, Triangle, and Ellipse Chart Drawing types (**DRAWING_RECTANGLEHIGHLIGHT**, **DRAWING_RECTANGLE_EXT_HIGHLIGHT**, **DRAWING_ELLIPSEHIGHLIGHT**, **DRAWING_TRIANGLE**), in order to see the outline it is necessary to set the LineWidth to 1 or greater. Otherwise, the outline will not be visible.

s_UseTool::LineStyle

Type: Integer

The line style for the chart drawing. This is an enumeration of type **SubgraphLineStyles**.

Available line styles are: **LINESTYLE_SOLID**, **LINESTYLE_DASH**, **LINESTYLE_DOT**, **LINESTYLE_DASHDOT**, **LINESTYLE_DASHDOTDOT**, **LINESTYLE_ALTERNATE**.

s_UseTool::Text

Type: [SCString](#)

The text to draw onto the chart. This applies to the **DRAWING_TEXT** tool. This member can also be used with the **DRAWING_HORIZONTALLINE** tool. In this case, this is the text that displays below the horizontal line.

For multiline text, each line needs to be terminated with a "\n" character.

If the Text member contains "\n" characters for multiline text, you also need to set

MultiLineLabel = 1 .

This string also has full support for [UTF-8](#). Therefore, all of the 1,000,000+ code points are supported. ASCII characters need to be encoded using 8-bit characters.

s_UseTool::DisplayHorizontalLineValue

Type: Integer

This member only applies to the DRAWING_HORIZONTAL tool. Set this to 1 to display the value of the horizontal line underneath the horizontal line. Set this to 0 to not display the value of the horizontal line. If this is set to any other value (default), then visibility of the horizontal line's value is determined using the setting under

Global Settings >> Tool Settings >> Horizontal/Vertical .

s_UseTool::FontSize

Type: Integer

The Font Size of the text. This applies to the DRAWING_TEXT tool.

s_UseTool::FontBackColor

Type: unsigned integer (COLORREF)

The color for the text background in [RGB format](#). This applies to the DRAWING_TEXT tool.

s_UseTool::FontFace

Type: [SCString](#)

The Font Face name. Such as Arial. This can be left unset to use the Font Face specified in **Global Settings >> Tool Settings >> Text** . This applies to the DRAWING_TEXT tool.

s_UseTool::FontBold

Type: Integer

Set this to 1 to make the font Bold. Set it to 0, to not. The default is 0. This applies to the DRAWING_TEXT tool.

s_UseTool::FontItalic

Type: Integer

Set this to 1 to make the font Italic. Set it to 0, to not. The default is 0. This applies to the

DRAWING_TEXT tool.

s_UseTool::TextAlignment

Type: Integer

This member can be optionally set to one or more of the following flags separated by the C++ OR (|) operator: **DT_TOP**, **DT_VCENTER**, **DT_BOTTOM**, **DT_CENTER**, **DT_LEFT**, **DT_RIGHT**. The default is **DT_TOP** and **DT_LEFT**. These alignment flags apply to the **DRAWING_TEXT** tool only and not any other tools.

This member can also be set to apply text alignment to the Text drawn on a **DRAWING_HORIZONTALLINE** drawing. In this case this can be set to **DT_LEFT** or **DT_RIGHT**. By default text is aligned to the left.

This member can also be a set to apply the text alignment to the Text drawn on a **DRAWING_RECTANGLEHIGHLIGHT** or **DRAWING_RECTANGLE_EXT_HIGHLIGHT** drawing. In this case, this can be set to **DT_LEFT** or **DT_RIGHT**. By default the text is aligned to the left.

s_UseTool::ReverseTextColor

Type: Integer

Set this to 1, to reverse the color of the text when using the **DRAWING_TEXT** drawing type. The text will be drawn with the chart background color and the area around the text will be drawn with the specified color ([Color](#) member).

The default value is 0, meaning a reversed color is not used.

This applies to the **DRAWING_TEXT** drawing type.

s_UseTool::MarkerType

Type: Integer

When using **DRAWING_MARKER**, this member is used to set the marker type to be drawn. The following constant values can be used:

- **MARKER_POINT**: Draws a point.
- **MARKER_DASH**: Draws a dash.
- **MARKER_SQUARE**: Draws a square.
- **MARKER_STAR**: Draws a star.
- **MARKER_PLUS**: Draws a "+".
- **MARKER_X**: Draws an "X".
- **MARKER_ARROWUP**: Draws an up arrow.
- **MARKER_ARROWDOWN**: Draws a down arrow.

- **MARKER_ARROWRIGHT**: Draws a right arrow.
- **MARKER_ARROWLEFT**: Draws a left arrow.
- **MARKER_TRIANGLEUP**: Draws a up triangle.
- **MARKER_TRIANGLEDOWN**: Draws a down triangle.
- **MARKER_TRIANGLERIGHT**: Draws a right triangle.
- **MARKER_TRIANGLELEFT**: Draws a left triangle.
- **MARKER_DIAMOND**: Draws a diamond.

s_UseTool::MarkerSize

Type: Integer

When using **DRAWING_MARKER**, this member is used to set the size of the marker be drawn, and should be greater than zero. This value is ignored if **UseBarSpacingForMarkerSize** is set.

s_UseTool::UseBarSpacingForMarkerSize

Type: Integer

When using **DRAWING_MARKER**, this member can be set to 1 to specify that the size of the marker drawn is set based on the bar spacing. When set, the marker size will scale with the bar spacing as it is changed.

s_UseTool::ShowVolume

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the total volume between the start and end points is displayed in the drawing result text.

s_UseTool::ShowAskBidDiffVolume

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the total Ask Volume minus the total Bid Volume between the start and end points is displayed in the drawing result text.

s_UseTool::ShowPriceDifference

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the price difference of the start and end points is displayed in the drawing result text.

s_UseTool::ShowTickDifference

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the price difference of the Begin and End points, measured in ticks, is displayed in the drawing result text. The Tick Size must be properly set for the chart.

s_UseTool::ShowCurrencyValue

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the price difference of the Begin and End points, measured in currency value, is displayed in the drawing result text. The Tick Size and Currency Value per Tick must be properly set for the chart.

s_UseTool::ShowPercentChange

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the price difference of the start and end points, measured as a percentage change, is displayed in the drawing result text.

s_UseTool::ShowTimeDifference

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the time difference of the start and end points is displayed in the drawing result text.

s_UseTool::ShowNumberOfBars

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the time duration of the start and end points, measured in number of bars, is displayed in the drawing result text.

s_UseTool::ShowAngle

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the angle of the calculator line is displayed in the drawing result text.

s_UseTool::ShowEndPointPrice

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the price value of the drawing endpoint is displayed in the drawing result text.

s_UseTool::ShowEndPointDateTime

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the date and time of the drawing endpoint is displayed in the drawing result text.

s_UseTool::ShowEndPointDate

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the date of the drawing endpoint is displayed in the drawing result text.

s_UseTool::ShowEndPointTime

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that the time of the drawing endpoint is displayed in the drawing result text.

s_UseTool::MultiLineLabel

Type: Integer

When using **DRAWING_CALCULATOR**, this member can be set to 1 to specify that result text should be split into multiple lines, one line for each result type. When not set, the result text is all contained on a single line.

When using **DRAWING_TEXT** and the **s_UseTool::Text** member contains new line characters ("\\n"), then MultiLineLabel needs to be set to 1 to cause the text to be displayed on multiple lines.

s_UseTool::ShowPercent

Type: Integer

When using **DRAWING_RETRACEMENT**, **DRAWING_EXPANSION**, or **DRAWING_PROJECTION**, this member can be set to 1 to specify that the level percentage be displayed in the drawing level text label.

s_UseTool::ShowPrice

Type: Integer

When using **DRAWING_RETRACEMENT**, **DRAWING_EXPANSION**, or **DRAWING_PROJECTION** this member can be set to 1 to specify that the level price value be displayed on the drawing as a text label.

When using **DRAWING_RECTANGLEHIGHLIGHT** or **DRAWING_RECTANGLE_EXT_HIGHLIGHT** this member can be set to 1 to specify that the price values be displayed as text labels on the rectangle drawings. In this case the alignment is controlled with [TextAlignment](#).

s_UseTool::RoundToTickSize

Type: Integer

When using **DRAWING_RETRACEMENT**, **DRAWING_EXPANSION**, or **DRAWING_PROJECTION**, this member can be set to 1 to specify that the level price values should be rounded to the nearest TickSize.

s_UseTool::ExtendLeft

Type: Integer

When using **DRAWING_RETRACEMENT**, **DRAWING_EXPANSION**, or **DRAWING_PROJECTION**, this member can be set to 1 to specify that the level lines will extend to the left.

s_UseTool::ExtendRight

Type: Integer

When using **DRAWING_RETRACEMENT**, **DRAWING_EXPANSION**, **DRAWING_PROJECTION**, **DRAWING_LINEAR_REGRESSION**, or **DRAWING_RAFF_REGRESSION_CHANNEL** this member can be set to 1 to specify that the lines will extend to the right.

s_UseTool::TimeExpVerticals

Type: Integer

When using **DRAWING_TIME_EXPANSION** or **DRAWING_TIME_PROJECTION**, this member can be set to specify the type of vertical time lines to draw. The possible constant values are:

- **TIME_EXP_EXTENDED**: The vertical time lines extend from the top to

bottom of the chart region.

- **TIME_EXP_STANDARD**: For **DRAWING_TIME_EXPANSION**, the vertical time lines extend from the drawing beginning price value to the drawing ending price value. For **DRAWING_TIME_PROJECTION**, the begin and end are drawing points 2 & 3.
- **TIME_EXP_COMPRESSED**: The vertical time lines are compressed around the horizontal time line.

s_UseTool::TimeExpTopLabel1

Type: Integer

When using **DRAWING_TIME_EXPANSION** or **DRAWING_TIME_PROJECTION**, this member can be set to specify the contents of the first top text label, which is placed above the corresponding vertical time line. The possible constant values are:

- **TIME_EXP_LABEL_NONE**: No value specified.
- **TIME_EXP_LABEL_PERCENT**: Time specified as percentage.
- **TIME_EXP_LABEL_BARS**: Time specified in number of bars.
- **TIME_EXP_LABEL_PERCENTBARS**: Percentage (Number of Bars).
- **TIME_EXP_LABEL_BARSPERCENT**: Number of Bars (Percentage).
- **TIME_EXP_LABEL_DATE**: Date for vertical.
- **TIME_EXP_LABEL_TIME**: Time for vertical.
- **TIME_EXP_LABEL_DATETIME**: Date-Time for Vertical

s_UseTool::TimeExpTopLabel2

Type: Integer

When using **DRAWING_TIME_EXPANSION** or **DRAWING_TIME_PROJECTION**, this member can be set to specify the contents of the second top text label, which is placed above the corresponding vertical time line. The possible values are specified above.

s_UseTool::TimeExpBottomLabel1

Type: Integer

When using **DRAWING_TIME_EXPANSION** or **DRAWING_TIME_PROJECTION**, this member can be set to specify the contents of the first bottom text label, which is placed below the corresponding vertical time line. The possible values are specified above.

s_UseTool::TimeExpBottomLabel2

Type: Integer

When using **DRAWING_TIME_EXPANSION** or **DRAWING_TIME_PROJECTION**, this

member can be set to specify the contents of the second bottom text label, which is placed below the corresponding vertical time line. The possible values are specified above.

s_UseTool::TimeExpBasedOnTime

Type: Integer

When using **DRAWING_TIME_EXPANSION** or **DRAWING_TIME_PROJECTION**, this member can be set to 1 to specify that the tool should use DateTimes instead of bar counts to calculate the time spans. This setting is really only relevant if the tool is being applied to a non-time based chart.

s_UseTool::TransparentLabelBackground

Type: Integer

For tools that use text labels and also the Text tools, this member can be set to 1 to specify that the labels/text should use a transparent background for the labels/text. Otherwise an opaque background is used.

However, in the case when the background color is not specified when using one of the Text tools, the background will be transparent.

s_UseTool::FixedSizeArrowHead

Type: Integer

When using **DRAWING_ARROW**, setting this member to 1 specifies that the **ArrowHeadSize** member specifies a fixed size. Otherwise, the **ArrowHeadSize** member specifies an arrowhead/body ratio.

s_UseTool::RetracementLevels[]

Type: float[32]

This array of float values specifies the levels for the drawing tools that require levels. These include **DRAWING_RETRACEMENT**, **DRAWING_EXPANSION**, **DRAWING_PROJECTION**, **DRAWING_TIME_EXPANSION**, **DRAWING_TIME_PROJECTION**, **DRAWING_PITCHFORK**, **DRAWING_PITCHFORK_SCHIFF**, **DRAWING_PITCHFORK_MODIFIED_SCHIFF**, **DRAWING_PARALLEL_LINES**, **DRAWING_PARALLEL_RAYS**, **DRAWING_LINEAR_REGRESSION**, and **DRAWING_RAFF_REGRESSION_CHANNEL**.

Each drawing tool has built in levels that define the basic tool. For example, **DRAWING_RETRACEMENT** has the initial angled line, and **DRAWING_PARALLEL_LINES** has the two base lines. The color, width, and style of these built-in levels are controlled with **Color**, **LineWidth**, and **LineStyle**.

For the other levels, the color width, and style are controlled with **LevelColor**, **LevelWidth**, and **LevelStyle**. All levels are specified as a floating point percentage (Example: **0.618**) except for **DRAWING_LINEAR_REGRESSION**, which specifies number of standard deviations (Example: **2.0**).

Using a value of **FLT_MAX** at a particular index in the **RetracementLevels[]** array means that level will not be drawn. Subsequently setting that level to any other value will restore the level to that particular value.

s_UseTool::LevelColor[]

Type: COLORREF[32]

This array specifies the colors for the corresponding tool levels set with **RetracementLevels[]**.

s_UseTool::LevelWidth[]

Type: int[32]

This array specifies the line widths for the corresponding tool levels set with **RetracementLevels[]**. The width needs to be greater than or equal to 1.

s_UseTool::LevelStyle[]

Type: int[32]

This array specifies the line styles for the corresponding tool levels set with **RetracementLevels[]**. Available line styles are: **LINESTYLE_SOLID**, **LINESTYLE_DASH**, **LINESTYLE_DOT**, **LINESTYLE_DASHDOT**, and **LINESTYLE_DASHDOTDOT**.

s_UseTool::AddAsUserDrawnDrawing

Type: Integer

This member specifies that a drawing is to be added as a user drawn drawing, which allows the user to interact with the ACSIL added drawing on the chart as if it had been added by the user with one of the drawing tools on the **Tools** menu.

Set this to 1 (TRUE) to make the drawing a user drawn drawing type. When a drawing is added with the **AddAsUserDrawnDrawing** member set to 1, this member must always be set to 1 when the drawing is updated/modified.

When working with ACSIL user drawn drawings, the **sc.UserDrawnChartDrawingExists()** and **sc.GetUserDrawnDrawingByLineNumber()** functions can be used to find and retrieve the drawing.

When you have added a chart drawing by setting **AddAsUserDrawnDrawing** to 1, then you must expressly delete the drawing by calling [sc.DeleteUserDrawnACSDrawing\(\)](#).

A chart drawing added with **AddAsUserDrawnDrawing** set to 1 is not saved to a Chartbook like an actual user drawn drawing would be.

s_UseTool::DrawUnderneathMainGraph

Type: Integer

When **DrawUnderneathMainGraph** is set to 1 (TRUE), then the Chart Drawing will be drawn underneath the main price graph and studies, independent of how the

Chart >> Chart Settings >> Chart Drawings >> Draw Non-Highlight Chart Drawings Underneath
or

Chart >> Chart Settings >> Chart Drawings >> Draw Highlight Drawings Underneath Main Graph
options are set.

When **DrawUnderneathMainGraph** is set to 0 (FALSE), then the chart drawing will be drawn above the main price graph and studies, independent of how the

Chart >> Chart Settings >> Chart Drawings >> Draw Non-Highlight Chart Drawings Underneath
or

Chart >> Chart Settings >> Chart Drawings >> Draw Highlight Drawings Underneath Main Graph
options are set.

When **DrawUnderneathMainGraph** is not set, then the chart drawing will be drawn based on the values of

Chart >> Chart Settings >> Chart Drawings >> Draw Non-Highlight Chart Drawings Underneath
and

Chart >> Chart Settings >> Chart Drawings >> Draw Highlight Drawings Underneath Main Graph
options are set.

The default for **DrawUnderneathMainGraph** is not set.

s_UseTool::UseToolConfigNum

Type: Integer

This can be set to 1 through 24 (Or whatever the maximum number of Chart Drawing Tool configurations currently supported in Sierra Chart is). This specifies the particular tool configuration defined for the Chart Drawing Tool specified, to automatically be applied to the Chart Drawing being added.

For additional information, refer to [Using Multiple Drawing Tool Configurations](#).

s_UseTool::FlatEdge

Type: Integer

Set this to 1 to use flat edges for the ends of a Line or Ray. This flag is mutually exclusive with the SquareEdge option.

s_UseTool::DrawWithinRegion

Type: Integer

Set this to 1 for Vertical Line drawings to cause them to be drawn with the specified Chart Region only instead of across all Chart Regions.

s_UseTool::CenterPriceLabels

Type: Integer

Set this to 1 for rectangle highlights to cause the price labels to be centered on the top/bottom price levels.

s_UseTool::NoVerticalOutline

Type: Integer

Set this to 1 for rectangle highlight drawings to cause the vertical outline to not be drawn. The result is a rectangle highlight with only top and bottom outline lines similar to a double extending rectangle highlight drawing.

s_UseTool::AllowSaveToChartbook

Type: Integer

Set this to 1 to allow a drawing added as a [User Drawn Drawing](#) to be saved to the Chartbook which contains the chart which contains the custom study which added the Chart Drawing, when the Chartbook is saved.

Set this to 0 to prevent saving of the Chart Drawing to the Chartbook.

s_UseTool::ShowBeginMark

Type: Integer

Set this to 1 to show a square mark at the beginning of a Line or other types of Chart Drawings which support marks at the anchor points. This is equivalent to the **Options >> Mark** setting in the [Drawing Tool Configuration/Properties](#) window.

s_UseTool::ShowEndMark

Type: Integer

Set this to 1 to show a square mark at the ending of a Line or other types of Chart Drawings which support marks at the anchor points. This is equivalent to the **Options >> Mark** setting in the [Drawing Tool Configuration/Properties](#) window.

s_UseTool::AssociatedStudyID

Type: Integer

AssociatedStudyID is set to a nonzero number when the Chart Drawing is associated with a **Volume by Price** study which was drawn on the chart using the [Draw Volume Profile](#) tool.

AssociatedStudyID specifies the unique ID for the study the Chart Drawing is associated with. This is equivalent to the ACSIL member variable [sc.StudyGraphInstanceID](#) for the study.

s_UseTool::HideDrawing

Type: Integer

When this is set to 1, it hides the Chart Drawing. Set this to 0 to make a hidden Chart Drawing visible.

s_UseTool::LockDrawing

Type: Integer

This member needs to be set to 1 to lock the chart drawing and prevent it from being modified through the chart user interface.

Set this to 0 or do not set it, to keep the drawing unlocked. It only makes sense to use this variable, for a user drawn drawing because only those types of drawings can be modified through the chart user interface.

s_UseTool::DrawOutlineOnly

Type: Integer

This member only applies to the following drawing types: DRAWING_TRIANGLE, DRAWING_MARKER, DRAWING_RECTANGLEHIGHLIGHT, DRAWING_ELLIPSEHIGHLIGHT, DRAWING_RECTANGLE_EXT_HIGHLIGHT.

When this is set to 1, it means the outline of the drawing will only be drawn. When this is set to 0 or is unset, then the interior of the drawing will also be drawn.

s_UseTool::NumCycles

Type: Integer

s_UseTool::ExtendMultiplier

Type: Float

s_UseTool::DrawMidline

Type: Integer

This member variable only applies to Chart Drawings which use a Midline option.

When this is set to 1, the Midline is drawn.

When this is set to 0 the Midline is not drawn.

s_UseTool::AllowCopyToOtherCharts

Type: Integer

When this is set to 1, it allows the Chart Drawing to be copied to other charts through the [Copy Chart Drawings](#) functionality.

When this is set to 0 or is unset, then the drawing cannot be copied to another chart. This is the default.

This only applies to user drawn drawings. This is when [s_UseTool::AddAsUserDrawnDrawing](#) is set to 1.

sc.ChartDrawingExists()

ACSIL Function

Declaration: int **ChartDrawingExists**(int **ChartNumber**, int **LineNumber**);

sc.ChartDrawingExists() is used to determine if a Chart Drawing with the specified **LineNumber** exists within the specified chart (**ChartNumber**).

This function ignores user drawn Chart Drawings, it only checks Chart Drawings added by ACSIL.

This function returns the number of matching drawings found. Refer to the **scsf_UseToolExample()** function in the /ACS_Source/studies.cpp file for example code on how to work with this function.

```
if (sc.ChartDrawingExists(sc.ChartNumber, 5))
{
    // A line with a LineNumber of 5 exists on the same chart that this study is on
}
```

sc.UserDrawnChartDrawingExists()

ACSIL Function

Declaration: int **UserDrawnChartDrawingExists**(int **ChartNumber**, int **LineNumber**);

sc.UserDrawnChartDrawingExists() function is used to determine if a user drawn Chart Drawing with the specified **LineNumber** exists within the specified chart (**ChartNumber**).

An ACSIL Chart Drawing is considered user drawn if it was added with the **s_UseTool::AddAsUserDrawnDrawing** member set to 1.

Non-user drawn ACSIL Chart Drawings are not searched with this function.

It returns the number of matching drawings found.

Refer to the **scsf_UseToolExample()** function in the /ACS_Source/studies.cpp file for example code on how to work with this function.

```
if (sc.UserDrawnChartDrawingExists(sc.ChartNumber, 5))
{
    // An ACSIL added user drawing with a LineNumber of 5 exists on the same chart that this study is on
}
```

sc.GetUserDrawnChartDrawing()

ACSIL Function

int **GetUserDrawnChartDrawing**(int **ChartNumber**, DrawingTypeEnum **DrawingType**, s_UseTool& **ChartDrawing**, int **DrawingIndex**);

sc.GetUserDrawnChartDrawing() is used to get a Chart Drawing drawn by a drawing Tool on the specified chart.

Only user drawn drawings can be retrieved, not Chart Drawings added by an ACSIL study, unless the Chart Drawing was specified as being **s_UseTool::AddAsUserDrawnDrawing** when adding it with **sc.UseTool()**.

This function returns 1 on success, and 0 on error. A error includes an invalid DrawingType specified or a drawing not found on the chart.

Refer to the **scsf_GetChartDrawingExample** function in the /ACS_Source/studies.cpp file for example code on how to work with this function.

Parameters

- **ChartNumber**: This number corresponds to the number after the number "#" sign on the top line of the chart. Using a zero (0) will specify the chart the ACSIL study is applied to.

- **DrawingType**: The chart drawing type that you want to get. This parameter can be any of the following.
 - DRAWING_LINE
 - DRAWING_RAY
 - DRAWING_HORIZONTALLINE
 - DRAWING_VERTICALLINE
 - DRAWING_ARROW
 - DRAWING_TEXT
 - DRAWING_CALCULATOR
 - DRAWING_RETRACEMENT
 - DRAWING_PROJECTION
 - DRAWING_RECTANGLEHIGHLIGHT
 - DRAWING_ELLIPSEHIGHLIGHT
 - DRAWING_TRIANGLE
 - DRAWING_FAN_GANN
 - DRAWING_PITCHFORK
 - DRAWING_CYCLE
 - DRAWING_TIME_EXPANSION
 - DRAWING_GANNGRID
 - DRAWING_ENTRYEXIT_CONNECTLINE
 - DRAWING_RECTANGLE_EXT_HIGHLIGHT
 - DRAWING_FAN_FIBONACCI
 - DRAWING_PARALLEL_LINES
 - DRAWING_PARALLEL_RAYS
 - DRAWING_LINEAR_REGRESSION
 - DRAWING_RAFF_REGRESSION_CHANNEL
 - DRAWING_EXTENDED_LINE
 - DRAWING_PITCHFORK_SCHIFF
 - DRAWING_PITCHFORK_MODIFIED_SCHIFF
 - DRAWING_EXPANSION
 - DRAWING_VOLUME_PROFILE
 - DRAWING_STATIONARY_TEXT
 - DRAWING_TIME_PROJECTION
 - DRAWING_MARKER
 - DRAWING_HORIZONTAL_RAY
 - DRAWING_HORIZONTAL_LINE_NON_EXTENDED
 - DRAWING_UNKNOWN. Use this constant to get any type of drawing. When the function returns and **DrawingType** was set to **DRAWING_UNKNOWN**, the **ChartDrawing::DrawingType** parameter member will indicate the type of drawing that has been found.
- **ChartDrawing**: A reference to a **s_UseTool** structure. This is where the Chart Drawing information will be copied to. For the descriptions of the **s_UseTool** structure, refer to [Drawing Tools and s_UseTool Member Descriptions](#).

User drawn Chart Drawings have a negative **LineNumber**. This is how you can distinguish Chart Drawings drawn by a user and also drawings flagged as user drawn which were added by the ACSIL.

- **DrawingIndex:**

An index to specify which instance of the matching chart drawing you want. This is a 0-based index in the order that chart drawings were added to the chart. For example, if **DrawingIndex** is 0, the function will return the first instance that matches the drawing type. If **DrawingIndex** is 1, the function will return the second instance that matches the drawing type.

Also you can give a negative index to specify drawings from the last drawing that was added to the chart. For example, if **DrawingIndex** is -1, the function will return the last drawing matching the drawing type that was added to the chart.

Be aware that when a Chart Drawing is modified, it is put at the end of the list. So in this case if you modify a Chart Drawing, it is going to be put at the end of the list, and be the one that is considered last added.

This index only counts the Chart Drawings on the chart that match the given **DrawingType**.

For example, if **DrawingType** is **DRAWING_RAY** and **DrawingIndex** is 1, then the function will return the second Ray on the chart. If **DrawingType** is **DRAWING_TEXT** and **DrawingIndex** is 1, then the function will return the second Text drawing on the chart.

If **DrawingType** is **DRAWING_TEXT** and **DrawingIndex** is -1, then the function will return the last Text drawing on the chart. If **DrawingType** is **DRAWING_UNKNOWN**, then **DrawingIndex** considers all Chart Drawings on the chart.

Example:

```
// Gets the last drawn text on the chart and adds a message to the Sierra Chart message log, if the text exists
s_UseTool ChartDrawing;
if (sc.GetUserDrawnChartDrawing(0, DRAWING_TEXT, &ChartDrawing, -1))
{
    sc.AddMessageToLog("Text was added on current chart!", 0);
    sc.AddMessageToLog(ChartDrawing.Text, 1);
}
```

sc.GetLineNumberOfSelectedUserDrawnDrawing()

ACSIL Function


```
int GetLineNumberOfSelectedUserDrawnDrawing();
```

The **sc.GetLineNumberOfSelectedUserDrawnDrawing()** returns the [Line Number](#) of the selected User Drawn drawing. If more than one drawing is selected, the first found drawing is returned.

sc.GetUserDrawnDrawingByLineNumber()

ACSIL Function

```
int GetUserDrawnDrawingByLineNumber(int ChartNumber, int LineNumber, s_UseTool& ChartDrawing);
```

The **sc.GetUserDrawnDrawingByLineNumber()** function is used to get a user drawn Chart Drawing on the chart. This also includes Chart Drawings added by ACSIL which are flagged as user drawn when setting **s_UseTool::AddAsUserDrawnDrawing** = 1 when the drawing is added with **sc.UseTool**.

This function cannot get Chart Drawings which are added by ACSIL when **s_UseTool::AddAsUserDrawnDrawing** = 0. To get non-user drawn Chart Drawings added by ACSIL, use the function [sc.GetACSDrawingByLineNumber\(\)](#).

This function returns 1 on success, and 0 on failure. A failure means the Chart Drawing was not found on the specified chart.

Refer to the **scsf_GetChartDrawingExample** function in the **/ACS_Source/studies.cpp** file for example code on how to work with this function.

When a user drawn Chart Drawing is being moved or adjusted by the user, the **sc.GetUserDrawnDrawingByLineNumber()** call will fail to find the Chart Drawing even though **sc.UserDrawnChartDrawingExists()** indicates that it exists.

Therefore, a study should always use **sc.UserDrawnChartDrawingExists()** to detect if the user drawn Chart Drawing exists, and then use **sc.GetUserDrawnDrawingByLineNumber()** to get the Chart Drawing, and understand that a failure means that the Chart Drawing is being modified.

When a Chart Drawing you are getting is not visible in the current view of the chart, then **s_UseTool::BeginIndex**, **s_UseTool::EndIndex**, **s_UseTool::ThirdIndex** will all equal -1.

Parameters

- **ChartNumber**: This number corresponds to the number after the number "#" sign on the top line of the chart you want to get the drawing from. Using a zero (0) will specify the chart the custom study is applied to. Usually set this parameter to 0 or **sc.ChartNumber**.
- **LineNumber**: This number corresponds to the **s_UseTool::LineNumber** of the ACSIL added user drawn Chart Drawing or the LineNumber retrieved when using the [sc.GetUserDrawnChartDrawing\(\)](#) function to get an actual user drawn drawing.

This parameter should not be 0 or -1.

- **ChartDrawing**: A reference to a **s_UseTool** structure. This is where the Chart Drawing information will be copied to. For the descriptions of the **s_UseTool** structure, refer to [Drawing Tools and s_UseTool Member Descriptions](#).

Note that when chart drawings are returned through this function, the **DrawingType** member is set to indicate the type of drawing retrieved.

sc.GetACSDrawingByLineNumber()

ACSIL Function

```
int GetACSDrawingByLineNumber(int ChartNumber, int LineNumber, s_UseTool& ChartDrawing);
```

The **sc.GetACSDrawingByLineNumber()** function is used to get an ACSIL added Chart Drawing on the chart added with the **sc.UseTool** function.

This function does not get user drawn Chart Drawings which are specified with **s_UseTool::AddAsUserDrawnDrawing = 1**. You need to use the [sc.GetUserDrawnDrawingByLineNumber\(\)](#) function instead.

This function returns 1 on success, and 0 on failure. A failure means the Chart Drawing was not found on the specified chart.

When a Chart Drawing you are getting is not visible in the current view of the chart, then **s_UseTool::BeginIndex**, **s_UseTool::EndIndex**, **s_UseTool::ThirdIndex** will all equal -1 unless the drawing specified its horizontal positioning using these *Index values when it was added.

Parameters

- **ChartNumber**: This number corresponds to the number after the number "#" sign on the top line of the chart. Using a zero (0) will specify the chart the custom study is applied to. Usually set this parameter to 0.
- **LineNumber**: This number corresponds to the **s_UseTool::LineNumber** of the ACSIL added Chart Drawing.

This parameter should not be 0 or -1. Otherwise, a match will not be found.

- **ChartDrawing**: A reference to a **s_UseTool** structure. This is where the Chart Drawing information will be copied to. For the descriptions of the **s_UseTool** structure, refer to [Drawing Tools and s_UseTool Member Descriptions](#).

Note that when chart drawings are returned through this function, the

DrawingType member is set to indicate the type of drawing retrieved.

sc.GetACSDrawingByIndex()

ACSIL Function

```
int GetACSDrawingByIndex(int ChartNumber, int Index, s_UseTool& ChartDrawing, int ExcludeOtherStudyInstances);
```

The **sc.GetACSDrawingByIndex()** function is used to get an ACSIL added Chart Drawing on the chart added with the **sc.UseTool** function. The drawing is looked up by its index number rather than by the **s_UseTool::LineNumber**.

This function returns 1 on success, and 0 on failure. A failure means the Chart Drawing was not found on the specified chart.

When a Chart Drawing you are getting is not visible in the current view of the chart, then **s_UseTool::BeginIndex**, **s_UseTool::EndIndex**, **s_UseTool::ThirdIndex** will all equal -1 unless the drawing specified its horizontal positioning using these *Index values when it was added.

Parameters

- **ChartNumber**: This number corresponds to the number after the number "#" sign on the top line of the chart. Using a zero (0) will specify the chart the custom study is applied to. Usually set this parameter to 0.
- **Index**: This zero-based index number is used to get a Chart Drawing according to its index in the internal list that ACSIL added Chart Drawings are added to.

The first Chart Drawing has an index of 0, the second Chart Drawing is 1 and so on.

Negative indexes are not supported.

- **ChartDrawing**: A reference to a **s_UseTool** structure. This is where the Chart Drawing information will be copied to. For the descriptions of the **s_UseTool** structure, refer to [Drawing Tools and s_UseTool Member Descriptions](#).

Note that when chart drawings are returned through this function, the **DrawingType** member is set to indicate the type of drawing retrieved.

- **ExcludeOtherStudyInstances**: When this is set to a nonzero value, then Chart Drawings added by other study instances are excluded. When this is set to 0, then Chart Drawings on the chart added by other study instances will be included when getting an Advanced Custom Study drawing by its index.

sc.GetACSDrawingsCount()

Type: ACSIL Function

```
int GetACSDrawingsCount(int ChartNumber, int ExcludeOtherStudyInstances);
```

The **sc.GetACSDrawingsCount** function returns the total number of ACSIL added Chart Drawings on the specified chart, specified by the **ChartNumber** parameter. This does not count user drawn ChartDrawings.

Parameters

- **ChartNumber:** [Type: Integer] The number of the chart that contains the drawing or drawings. The default and recommended value is 0, which means the chart the ACSIL study instance is applied to. This number corresponds to the number after the number "#" sign on the top line of the chart.
- **ExcludeOtherStudyInstances::** When this parameter is set to 1, then Chart Drawings from other study instances on the chart is excluded in the drawing count. When this is set to 0, all Chart Drawings added by all custom studies are counted.

sc.DeleteACSChartDrawing()

ACSIL Function

```
Declaration: int DeleteACSChartDrawing(int ChartNumber, int Tool, int LineNumber);
```

The **sc.DeleteACSChartDrawing()** function deletes Chart Drawings that have been added by ACSIL studies using the [sc.UseTool\(\)](#) function.

This function cannot be used to delete user drawn Chart Drawings. User drawn Chart Drawings are manually drawn by a user or ACSIL added Chart Drawings that have the **s_UseTool::AddAsUserDrawnDrawing = 1** flag set. For deleting user drawn drawings, use [sc.DeleteUserDrawnACSDrawing\(\)](#).

When a study is removed from the chart, or when it is fully recalculated, and it has Chart Drawings added with [sc.UseTool\(\)](#), these Chart Drawings are automatically deleted from the chart. Therefore, there is no need to use **sc.DeleteACSChartDrawing()**.

This function should only be used for more specialized purposes like when you need to delete a Chart Drawing for some other reason.

For an example which uses this function, refer to the **scsf_DeleteACSChartDrawingExample** function in the **/ACS_Source/studies.cpp** file in the Sierra Chart installation folder.

Parameters

- **ChartNumber:** [Type: Integer] The number of the chart that contains the

drawing or drawings to delete. The default and recommended value is 0, which means the chart the ACSIL study instance is applied to. This number corresponds to the number after the number "#" sign on the top line of the chart.

- **Tool:** [Type: Integer] The **Tool** parameter can be **TOOL_DELETE_ALL** or **TOOL_DELETE_CHARTDRAWING**.

If Tool is set to **TOOL_DELETE_ALL**, then all non user drawn Chart Drawings on the specified **ChartNumber** will be deleted. The **LineNumber** parameter in this case is ignored and should be 0.

If Tool is set to **TOOL_DELETE_CHARTDRAWING**, then all non user drawn Chart Drawings with the specified **LineNumber** will be deleted from the chart specified by the **ChartNumber** parameter.

- **LineNumber:** [Type: Integer] The LineNumber of the Chart Drawings to delete. The LineNumber is the same **LineNumber** that was specified when adding the Chart Drawing with **sc.UseTool()**. In the case where LineNumber was not set when calling **sc.UseTool()**, then it will be automatically set. In this case remember it into a [persistent integer](#) and then it can be specified in this function call.

This parameter only applies to **TOOL_DELETE_CHARTDRAWING**.

Multiple Chart Drawings can be deleted if they use the same **LineNumber**. All chart drawings with the same **LineNumber** will be deleted from the chart.

Return Value

For **TOOL_DELETE_ALL**, 1 on success, 0 on failure.

For **TOOL_DELETE_CHARTDRAWING**, the number of Chart Drawings deleted.

sc.DeleteUserDrawnACSDrawing()

ACSIL Function

Declaration: int **DeleteUserDrawnACSDrawing**(int **ChartNumber**, int **LineNumber**);

The **sc.DeleteUserDrawnACSDrawing()** function deletes Chart Drawings that have been added by ACSIL studies using **sc.UseTool()** and have specified

s_UseTool::AddAsUserDrawnDrawing = 1.

Chart Drawings manually drawn by a user can also be deleted with this function if the **LineNumber** for the drawing is known. This can be determined with the [sc.GetUserDrawnChartDrawing](#) function.

When Chart Drawings have been added **sc.UseTool()** and have specified **s_UseTool::AddAsUserDrawnDrawing = 1**, it is necessary to remove these drawings with

sc.DeleteUserDrawnACSDrawing() in the study function when **sc.LastCallToFunction** is TRUE. Otherwise, they will remain on the chart.

When **sc.DeleteUserDrawnACSDrawing()** successfully removes a user drawn Chart Drawing, the internal storage container will no longer contain this Chart Drawing. Therefore, the index position of Chart Drawings after this deleted Chart Drawing in the container, changes. Therefore, the index specified to functions like [sc.GetACSDrawingByIndex\(\)](#) may be affected by this.

Parameters

- **ChartNumber**: [Type: Integer] The number of the chart that contains the Chart Drawing. The default value is 0, this means the chart the ACSIL study is applied to. This number corresponds to the number after the number "#" sign on the top line of the chart.
- **LineNumber**: [Type: Integer] The **LineNumber** of the Chart Drawing to delete. The **LineNumber** of the Chart Drawing must be known.

Since user drawn chart drawings each need to have a unique **LineNumber**, when the first user drawn drawing encountered has a matching **LineNumber**, it is deleted and no other drawings will be checked.

The **LineNumber** is the same **LineNumber** that you specified when adding the chart drawing with the [sc.UseTool](#) function, or was automatically set and remembered.

This can also be a **LineNumber** from an actual user drawn Chart Drawing retrieved with the [sc.GetUserDrawnChartDrawing](#) function.

Return Value

The number of user drawn Chart Drawings deleted.

RGB Color Values

Colors are stored as 32-bit unsigned integer values. It is easiest to use the RGB function. It takes red, green, and blue integer values as parameters, and returns a color value. For more information, you may want to refer to the [RGB color model](#) Wikipedia article.

Here are some color examples:

- Red: RGB(255,0,0)
- Green: RGB(0,255,0)
- Blue: RGB(0,0,255)
- Magenta: RGB(255,0,255)
- White: RGB(255,255,255)
- Black: RGB(0,0,0)

- **Dark Blue**: RGB(0,0,127)
- **Orange**: RGB(255,127,0)

```
sc.Subgraph[0].PrimaryColor = RGB(255, 0, 0); // Set the primary color for the first subgraph to red
s_UseTool Tool;
Tool.Color = RGB(0, 0, 255); // Set Tool color to Blue
```

Drawing Chart Drawings On Top of or Underneath Main Graph and Studies

The Chart Drawings added by the **sc.UseTool** function can be set to be displayed on top of or underneath the Main Price Graph and Studies in the chart. This can be done by using the [DrawUnderneathMainGraph](#) member of the [s_UseTool](#) structure.

Also, there are 2 settings for this as well in **Chart >> Chart Settings >> Chart Drawings**.

They are **Draw Non-Highlight Chart Drawings Under Main Graph and Studies** and **Draw Highlight Drawings Under Main Graph and Studies**.

Adding Chart Drawings to Other Charts from an ACSIL Study Function

It is supported when using the [sc.UseTool\(\)](#) function, to add Chart Drawings to a chart other than the chart the study function making the function call is applied to.

This can be done by setting the [s_UseTool::ChartNumber](#) member to the chart number that you want to add a drawing to.

It is also necessary to set [s_UseTool::AddAsUserDrawnDrawing](#) to 1.

Drawing Lines With a Specific Angle Using ACSIL

This section explains how to use ACSIL to draw lines at a specific angle. For introductory information about using drawing tools from ACSIL, refer to the [Introduction](#) section on this page.

The **s_UseTool** structure supports specifying the beginning and ending anchor points of the line using these members:

- [BeginDateTime](#)
- [EndDateTime](#)
- [BeginIndex](#)
- [EndIndex](#)
- [Begin Value](#)
- [End Value](#)

You need to make sure the combination of these values results in a line with a specific angle that you require. For example a 45 degree line will have a slope of 1. This means one unit of price over one unit of time. One unit of time is always one bar in the chart. The [Drawing a Line with a Specific Angle or Slope](#) section has complete information on this subject.

To have a better idea of all of this, it is recommended to manually draw [Lines](#) at 45 degrees to get an idea of what values to use.

*Last modified Wednesday, 05th July, 2023.