Contents

Thank you for choosing Chart FX 3.0 as your charting tool for your Windows Application...

Press the Contents Button To Activate Smart Help Display.

General Topics

Programmer's Guide

For Help on Help, Press F1

General Topics. Overview

Once the chart is created and you already passed information to it (as explained in the Passing Data section), you can interact with the chart by handling properties, either at design or run time.
In the following section we will cover only general topics important to further customize the charts using those properties.

If you are in the stage of further customizing the chart we strongly suggest that you jump to the Programmers Guide Samples section for information on more general topics and samples.

If you are already familiar on how to interact (programatically) with the chart we suggest you use the Quick Reference card to locate the properties or messages or go directly into the Function & Properties reference of this electronic help file.

If youre not yet familiar with all the features of the library, or do not understand the application of certain properties, we strongly suggest that you show all end user tools available in Chart FX and interact with the chart by setting features at running time. This will give you an idea of what you can accomplish with the library. Once youre familiarize with what you can do, you can start looking at the Properties reference of this manual to set those features programatically.

Have Fun!

Passing Data. Communication channels

In order to open and close a communication channel using the VBX-OCX you must use the OpenData and CloseData properties respectively. The prototypes of these functions list as follows:

```
[form.] Chart1.OpenData(Index) = setting&
[form.] Chart1.CloseData(Index)  =  0
```

These two properties must be used when you want to certain information to the chart. The Index is specified by a COD_ constant that will allocate the appropiate amount of memory for the items you want to pass to the library.
For every communication channel opened you must close it after setting the appropiate information. Also note that the OpenData property has a setting that will allocate the exact amount of memory that you need for the different items in your chart.

A pseudo-code to open and close a communication channel using the VBX-OCX will look like:

```
 Open the communication channel to input data*/
Chart1.OpenData($DataType) = nData
 Control Structure to fill the data. In C code */
For j=0 to 20 step 1
       .
       /* Obtain the value you want to graph */
       /* Set the value */
       .
}
/* Close data communication channel */
Chart1.CloseData($DataType) = 0
```

Please note that the CloseData property is called after setting the appropiate information to the chart. In the following pages we will show you how to fill each parameter of the OpenData and CloseData properties depending on the information you want to send to the chart.

Passing Data. Data Types Reference

As we mentioned before, there are several types of dynamic data you may pass to the chart. To open the communication channel for each data type you must include the following constants in the second parameter of the Index in the OpenData property and specify the same constant in Index of the CloseData property to close this opened channel.
The following table also describe the property or function that you have to use in order to set such item in the chart:

Code	**Description**	**VBX-OCX properties**
COD_VALUES	This constant is used to open a channel that will pass the values you want to plot in the chart.	**Value property** ThisSerie Property
COD_INIVALUES	This constant is used to open a channel that will pass the values you want the bars to be initialized to when creating a Bar Chart. This constant is applicable only to this type of chart, and the effect of applying these ini values is that each bar of any series can start in some point other than zero, achieving a Gantt style chart.	**IniValue property** ThisSerie property
COD_XVALUES	This constant is used to open a channel that will pass the values you want to plot in the x-axis of a Scatter chart. This constant is applicable only to this type of chart	**Xvalue property** ThisSerie property
COD_CONSTANTS	This constant is used to open a channel that will pass the horizontal constant values you want to plot in the chart.	**Const property.**
COD_COLORS	This constant is used to open a channel that will pass the colors you want for each series plotted in the chart	**Color property** ThisSerie property
COD_STRIPES	This constant is used to open a channel that will pass the stripes or color frames you want to paint in the chart, as a way to denote a specific zone in the chart.	**chart_SetStripe function** *** Not supported as a property
COD_STATUSITEMS	This constant is used to open a channel that will pass the status bar items	**chart_SetStatusItem function** *** Not supported as a property

Passing Data. Special Cases

Note that COD_VALUES, COD_XVALUES and COD_INIVALUES must have the same size for the type of chart you are making. i.e If you are making a bar chart with 10 points and want each bar to start at certain point, you must allocate the same size (10 points) in both COD_VALUES and COD_INIVALUES OpenData property call. Equally, if you are making a scatter chart you must pass the same number of points in the COD_VALUES and COD_XVALUES.

On the other hand, if you are accessing information from a database or from a media that does not permit you to know the amount of points to be plotted in the chart, Chart FX can solve this problem by allocating memory dynamically in accordance with the amount of points you are passing to it in the control structure. To handle these cases, Chart FX supports two codes (or constants) that you can combine in the CHART_ML macro (just for COD_VALUES,COD_XVALUES and COD_INIVALUES datatypes) to assure that all the memory allocated has the same size and prevent inconsistency of the data. These constants are:

Code **Description**

COD_UNKNOWN Is used to specify that you do not know the amount of points you are going to pass to the chart in a OpenData property call. This will cause the library to allocate memory dinamically in accordance with the amount of points you're passing to it (generally in the control structure, where you pass the data).
See Sample Case 1

COD_UNCHANGE Is used to specify that you want to keep actual number of series and points (i.e. you want to change only one value in a previously created chart).
See Sample Case 2

Sample Case 1:
Suppose you want to create a bar chart with 2 series and the number of points unknown (because they come from a database, and you do not want to count the number of records first).

```
' Open the channel with an unknown number of points
Chart1.OpenData(COD_VALUES) = CHART_ML(2,COD_UNKNOWN)
' Control Structure to fill the data. In pseudo code
i=0
while not eof() {
       ' obtain the value from the database
       read record
       fValue=field_3
       ' Set the value by calling the Value property
       i=i+1
}
' Close data communication channel
Chart1.CloseData(COD_VALUES)=0
```

Sample Case 2:
Suppose you want to change the third value of the first serie of a previously created chart.

```
'Chart is already created and handle stored in hGraph Variable
' Open the channel to modify a value
Chart1.OpenData(COD_VALUES) = CHART_ML(COD_UNCHANGE,COD_UNCHANGE))

' Remember this numbers are zero based
```

```
Chart1.ThisSerie = 0
Chart1.Value(2) = 45.67

'here you can modify any number of values

' Close data communication channels
Chart1.CloseData(COD_VALUES)=0
```

Passing Data. Detecting Memory changes

When you access the OpenData property or use any of the functions or properties provided to pass information to a chart, Chart FX return several codes that can be useful for several purposes. These constants are returned in those functions in order to notify the programmer of any changes made in the size of memory allocated.

These constants (returned values) are:

Returned Value	**Description**
CR_FAIL	This constant is returned when OpenData property can not allocate the necessary memory to create the chart. We suggest that you always check for this return value when calling this function.
CR_NEW	This constant is returned when OpenData property is called for the first time with a specific data type (i.e. COD_VALUES). With this flag you can determine if you passed this data type previously to the chart.
CR_SERIELOSS	This constant is returned when OpenData property is called in order to resize the previously allocated memory and one or more series are lost because of that change.
CR_KEEPALL	This constant is returned when OpenData property is called in order to resize the previously allocated memory and all the data passed to the chart is maintained. This might happen when you increase the number of series or points you had in the chart.

The constants (return values) for the several set functions provided are:

Returned Value	**Description**
CR_SUCCESS	This constant is returned when the set function passed the data successfully to the chart.
CR_NOOPEN	This constant is returned when the set function is called without making a OpenData property call. We suggest you always check for this return value.
CR_OUTRANGE	This constant is returned when the set function tries to set a point or series outside the range specified in the OpenData property

Handling Notification Messages. Overview

Notification messages (events) are the standard way child windows inform parent windows of changes and related information, this is the way all controls (ListBoxes, ComboBoxes, Edit Controls, etc.) work in Microsoft Windows and is also how CHART FX works.
The events supported in Chart FX allow you to capture end user actions in the chart so you can change default processing and give your application a special way to handle them. For example, capturing the double-click event in any marker of the chart will allow you to display any text within the default balloon help or even route your application to an specific module that handles such information, you can also capture special events to customize chart drawing and place your own objects in the chart.

Notification messages are sent through events to the chart control. Each event has its own parameters or information regarding that message. These parameters will allow you to change how Chart FX processes these events by default.
All you have to do is process these events and add your own code to change how Chart FX normally behaves with each one.
An important issue is for those events that you as a programmer can stop default processing. for such events we have included the nRes parameter that you have to set to 1 to force Chart FX to stop processing such event the usual way

Handling Notification Messages. Notification Codes

The supported notification messages in the VBX-OCX are listed in the following table. Please note that some of them have to be used in conjunction with the other related properties for additional information regarding that event. Also another column named Other properties contains other messages related to such event. Also in the table you will find a description of all the parameters included in the event.
Also remember that those events with the last parameter specify as nRes (Integer) can be stop by assigning 1 to this variable. If you fail to do this (nRes =1) Chart FX will continue default processing and the code you place in any of this events will take no effect.

Chart Supported events are:

ChangeColor event	ChangeFont	ChangePalette	ChangePattern
ChangePattPal	ChangeString	ChangeType	ChangeValue
Destroy event	GetLegend	GotFocus	InternalCommand
LButtonDblClk	LButtonDown	LButtonUp	LostFocus
Menu	MouseMove	Paintmarker event	PostPaint
PrePaint	RbuttonDblClk	RButtonDown	RButtonUp
ReadFile	ReadTemplate	ShowToolbar	UserCommand
UserScroll			

Customizing the Toolbar. Overview

Chart FX 3.0 Toolbar can be customized to change its appearance, to show new buttons or even client application propietary child windows. Since the objects that you can display in the toolbar are no longer buttons we call these objects "Toolbar Items", since all of them may not be buttons.

Chart FX 3.0 Toolbar supports up to 32 items that are already created when you show/hide the Toolbar. This will make the process of customizing a lot easier since all you do is changing items properties instead of adding new ones.

When you change items (i.e. displaying a propietary Combobox) the Toolbar will automatically adapt its width to fit the new items also supporting dockable capabilities (Floating Toolbar).

In order to provide customization support we have documented all existing button IDs and give you a complete description on how to customize the toolbar. Since customizing the toolbar is a feature that can be used for many purposes we would like you to know the general steps to customize Chart FX 3.0 Toolbar.

1) The first step to customize the Toolbar is to know which of the existing items you want to change. Most left item 0.

2) Assign the item ID with the TblTemId property, which specify how Chart FX will handle that item (even if it is Chart FX propietary item or your custom item).

3) Assign the item Style with the ItemStyle property , which will specify what kind of control you want to place in the toolbar.

4) Redesign the Toolbar Picture to fit customized or new buttons in the Toolbar.

5) Handling the events for custom items added to the Toolbar.

Important Note: Chart FX 3.0 display a new and enhanced Toolbar that will give your end-users access to the most powerful tools in the library. No additional customization is needed to display this default toolbar.

Next:
What can you display?
Changing Toolbar Items
Changing Toolbar Picture
Handling Toolbar Events
Enabling/Disabling Toolbar Items
Show/Hide Toolbar Items
When to customize the Toolbar?

Customizing the Toolbar. What can you display?

Chart FX 3.0 supports differents items that can be passed and display in the Toolbar. Since Chart FX 3.0 Toolbar support customization the programmer can display up to 32 different items in it. All the items are already created when you display the Toolbar (some of them hide by default), which means that you will not add new items but instead changing the existing ones. For more information on customizing the toolbar please refer to "Changing Toolbar Items" later in this section.

Every item has its position and ID (except for separators!) in the Toolbar, this means that you can assign a position and an ID number to process every time the end users interact with each item. For more information related on this topic, please refer to "Handling Toolbar Events" later in this section.

The items supported by Chart FX 3.0 are:

Buttons: These represent standard Windows buttons that can be click by the end user to generate an action in Chart FX or in the client application. Each Chart FX 3.0 button display a picture or bitmap inside of it (usually 16x15 pixels) and they can exist individually or in groups in the toolbar. The different buttons that can be display are:

· **1-State buttons**: This is a standard Windows push button, with a picture inside of it.
· **2-State buttons**: this a button that remains pushed until the end-users click on another button in the same group or on it again.
· **Menu buttons**: This kind of buttons will display a menu when the end-users click on them.
· **Timer Buttons**: Represent buttons that end-user can keep pressed to generate a burst of events.

Separators: represent an empty space in the toolbar with an specific width (in pixels) to separate individual or button groups.

Icon Combos (Graphic) : These are new type of items that list a graphic representation (icons) of the selection that can be choose by the end-users. Chart FX 3.0 gallery and colors are displayed using these items. Programmers can not create these type of items since they are private to Chart FX 3.0 gallery and color selection. Nevertheless, you can assign a position or show/hide these special items in the Toolbar.

User control: Any child window (Windows standard controls such as: comboboxes, edits, check buttons, radio buttons,etc) can now be placed in Chart FX 3.0 Toolbar.

For more information on how to use every item supported please refer to "Changing Toolbar Items" later in this section.

IMPORTANT NOTE: IF YOU WANT TO CUSTOMIZE THE TOOLBAR YOU MUST APPLY ALL CHANGES TO IT AS EXPLAINED IN When to Customize the Toolbar LATER IN THIS SECTION

Customizing the Toolbar. Changing Toolbar Items

In Chart FX 3.0 every Toolbar item have a style and depending on this style it also has another integer that could be the ID or other pertinent information to that item.Remember: All 32 items are already created in the Toolbar, you as a programmer can change the ID and the style of any of the 32 items of the Toolbar. No more than 32 items can be display in Chart FX 3.0 Toolbar.
Important: All settings explained in this topic must be placed in an specific part of your source code explained in the "When to customize the Toolbar" section later in this chapter.

The process of Changing an Item in the Toolbar is as follows:
1) Know the index of the button that you want to change in the toolbar. From 0 to 31, where zero represents the most left item in the toolbar.

2) Setting the ID of the Item.

3) Setting the style for the item.

Important note: The ID must be set before setting the item style.

SETTING THE ITEM ID

An ID must be associated to that item in order to work properly. The following table explain the meaning of ID for every type of items supported:

Item	**Style**	**ID Setting**
1-State	CTBS_BUTTON	The ID that the button will generate when pressed.
2-State	CTBS_BUTTON CTBS_2STATE	The ID that the button will generate when pressed.
Menu Buttons	CTBS_BUTTON CTBS_MENU	Menu Handle or a predefined Chart FX button menu ID.
Timer Buttons	CTBS_BUTTON CTBS_REPEAT	The ID that the button will generate while pressed.
Icon Combo	CTBS_ICONCOMBO	n ID of any of the predefined Icon Combo.
User Controls	CTBS_HWND	Handle of the Window.
Separator	CTBS_SEPARATOR	Separation Width (in pixels)

PREDEFINED IDS IN CHART FX 3.0

Since Chart FX 3.0 Toolbar (and menu) has default functions and items (buttons) you must know each ID predefined to default buttons, so you can reassign or rearrange the buttons keeping the original Chart FX Toolbar functionality. Also, knowing these IDs will allow you to assign other IDs to your propietary so they do not interfere with internal commands. See Also Toolbar pre-defined items.

Existing items in the Toolbar.
To assign a pre-defined ID to an item in the Toolbar with the purpose of keeping the original item functionality you can use any of the CFX_ID_ constants. See Also Toolbar pre-defined items.

Propietary buttons.
Since you can change an existing item in the toolbar and assign to it a new style and ID to handle an specific function in your client application you must be aware not to assign an ID that handles a pre-define

function in Chart FX.
To make this easier we have define two constants CFX_ID_FIRST and CFX_ID_LAST which define the range in which Chart FX 3.0 public IDs are defined. Therefore, if you are to handle propietary items in your application you must use an ID lesser than CFX_ID_FISRT.
Important Note: Values greater than CFX_ID_LAST are private. Therefore, they can not be assigned to any Toolbar Item. Assigning a value greater than CFX_ID_LAST as an ID for a Toolbar ID will cause unpredictable behavior of the library.

IMPORTANT NOTE: If you are going to change the existing items of the toolbar to support your own functions in your client application, or for any reason change the picture used in any of the buttons of the Toolbar. Please refer to "Changing Toolbar Picture" later in this section.

Setting the Item ID using the VBX-OCX
The property related to set the item ID in the VBX model is TbItemID. This property is used to set/get the ID of an item in the toolbar.The Prototype for this property is:

```
Visual Basic
[form.] Chart1.TbItemID(Index) [ = setting$ ]
```

Sample

```
Chart1.TbItemID(1) = CFX_ID_GALLERY
```

SETTING THE ITEM STYLE

The possible styles that represent the different items explained in the What can you display ? section are:

Style	Hex Value	Description
CTBS_BUTTON	0x0001	Button
CTBS_MENU	0x0004	Menu Button. Must be used with CTBS_BUTTON
CTBS_2STATE	0x2000	2-State Button. Must be combined with CTBS_BUTTON
CTBS_REPEAT	0x0800	Timer Button. Must be used with CTBS_BUTTON.
CTBS_HEAD	0x4000	Identify the first button of a group.
CTBS_GROUP	0x1000	Identify the button is in the current group
CTBS_GROUP2STATE	0x3000	Combination of CTBS_2STATE and CTBS_GROUP.
CTBS_SEPARATOR	0x0002	Separator (Blank Space)
CTBS_ICONCOMBO	0x0008	Icon Combo (Only for Gallery Type and Color Palette).
CTBS_HWND	0x0088	User Controls
CTBS_DESTROY	0x0010	Combined with CTBS_HWND specify that Toolbar will destroy the control when the toolbar is show or hide.

Setting the Item Style using the VBX-OCX
The property related to set the item ID in the VBX model is TbItemStyle. This property is used to set/get the style of an item in the toolbar.The Prototype for this property is:

```
Visual Basic
[form.] Chart1.TbItemStyle(Index) [ = setting$ ]
```

Sample

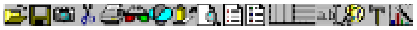
To convert the first item of the toolbar to be a push button

```
Chart1.TbItemStyle(0) = CTBS_BUTTON
```

Customizing the Toolbar. Changing Toolbar Picture

Since Chart FX allows you to change and customize the Toolbar adding new and custom buttons to your applications the Toolbar Picture must be a handle to a bitmap (HBITMAP) that contains:

The same number of icons as buttons you have in the Toolbar, of 16 pixels in Width by 15 pixels in Height (16x15) joined without any space between them. i.e.
Chart FX 3.0 Toolbar Picture look like:



How Chart FX process this Picture?
This picture must have the same number of icons as your Toolbar have, it does not matter the position of each button in the Toolbar, the first icon correspond to the first button that appears in the Toolbar, not matter if this button has other position than the first in your Toolbar.

Setting the Toolbar Picture using the VBX-OCX
The property related to change the Toolbar Picture VBX model is TbBitmap. This property is used to set the new toolbar picture.
The Prototype for this property is:

```
Visual Basic
[form.] Chart1.TbBitmap [ = setting$ ]
```

```
'To set a new Toolbar Picture
Chart1.TbBitmap = LoadPicture("c:\chartfx3\newtool.bmp")
```

Customizing the Toolbar. Handling Toolbar events

Chart FX will generate an event every time the end users interact with any of the items contained in the Toolbar. This event depends on the ID that you have assigned to the item. The different events are described as follows:

CHART FX PUBLIC PRE-DEFINED IDS. (FROM CFX_ID_FIRST TO CFX_ID_LAST)
Whenever the user clicks (or interact) with any item on this category Chart FX will generate a notification message with the following information:

Handling Default items using the VBX-OCX

An event called InternalCommand receiving two parameters:

Parameter	Description
wParam	contains the ID of the command (CFX_ID_)
lParam	Not used

This notification is generated so if programmers want to handle this event in a particular way in the client application they may capture it and process it thereby canceling the default library behavior for this item in the Toolbar. Returning 0 when receiving this notification means that Chart FX will continue with default processing. Otherwise Chart FX will not performed as expected.

CLIENT APPLICATION PROPIETARY ITEMS
Whenever the user make any action (or interact) with any item contained in this category, Chart FX will generate a standard windows message with the following information:

Handling Propietary items using the VBX-OCX

An event called UserCommand receiving two parameters:

Parameter	Description
wParam	contains the ID assigned to that item
lParam	contains specific information

Customizing the Toolbar. When to customize the Toolbar?

Since end-users can choose whether to show or hide the Toolbar, you must customize the Toolbar each time the end users show it (Since Chart FX destroys the Toolbar when is hidden). For this purpose Chart FX 3.0 sends you a notification in which you have to place all source code to customize the Toolbar.

When to Customize the Toolbar using the VBX-OCX
The ShowToolbar event is sent every time the Toolbar is created

For more information on how to handle standard Chart FX messages, please refer to "Handling notification Messages" in your Chart FX 3.0 help file.

Enabling/Disabling Toolbar Items

Chart FX 3.0 allows you to enable/disable any of the items contained in the Toolbar, no matter if it is a button or a propietary control.

Enabling/Disbaling Items using the VBX-OCX
The property related in the VBX model is EnableTbItem. This property is used to enabling/disabling toolbar items.
The Prototype for this property is:

```
Visual Basic
Chart1.EnableTbItem(Index) = seeting&
```

'To disable the third item in the Toolbar

```
Chart1.EnableTbItem(2) = FALSE
```

Important Note: If you want to Show/Hide items in the Toolbar, please refer to: CustomTool and GalleryTool properties

RealTime Charts. OverView

Chart FX 3.0 supports True Realtime Charting capabilities, by giving specific functions which support Chart scrolling in a very fast painting mode (without flickering). To prevent data overflow Chart FX 3.0 introduces Real Time Charts with an specific maximum number of points which will allow the library to accept up to that number and after you insert or set a new value to the chart the first set value is lost.

Chart FX supports two different Real Time Charts:

1) **Limited Real Time**: Which are charts that have a maximum values of points (previously allocated buffer). This type of charts are the fastest available in Chart FX, since they allocate memory only once (when you call MaxValues property). which means that the chart will have the maximum setting until it begins to lose points (i.e. setting 15 as the maximum number of points means that you will lose point 1 when passing value for point 16). If losing previously passed points is not important we suggest you use this type of charts for Real Time purposes when having a fast data input rate.

Two variations of this kind of chart are also available in Chart FX:

- **Standard**: When the buffer is full (You have reach the max value limit) and you insert new point, the data will "scroll" so you will lose the first point and the nth point will become the nth-1.
- **Loop Position:** Same as Standard but every time you set a new value a customizable vertical line will pass through the point that is being changed "Last acquired point" when reaching the end of the data set this Looping marker will move to the beggining

2) **Unlimited Real Time Charts**: These type of charts can add points to the existing ones without losing any of the previous ones up to the limitations imposed by the 64K data segment in Chart FX. Also, you may choose if the chart will scroll every time it receives a new point, so you can see the last acquired data. This is to allow you to set if the chart will automatically scroll depending on the context of your application.

Next:

How to Create and pass data to a Real Time Chart?
Setting the Real Time Style.
Customizing the Loop Marker

RealTime Charts. Creating and passing data

Creating a RealTime Chart using the VBX-OCX

1) Set the CT_EVENSPACING to the chart type using the Type property (either at design or running time).

2) Set the MaxValues property to an specific number of points if you want to create a Limited RealTime chart (strongly suggested!).

3) call the RealTimeStyle property to select the Real Time Style you want (Basically showing or not the Loop marker and hiding the hourglass cursor).

4) Open the communication channel to the RealTime Chart using the COD_ADDPOINTS combined with COD_VALUES. This will cause the pointers to the data array be relative to the last point added previously, so you don't have to remember neither the number of points the chart currently has nor the index of the last point passed.

5) Set the corresponding value of the new points using an offset instead of an absolute index to the points. These means that if you want to add two (2) new points to the chart (before the CloseData property) you must use index 0 and 1 for the point index in the Value property.

6) Call the CloseData property with a combination of COD_VALUES   with any of the following constants:
COD_REALTIME Chart FX will not scroll to the end of the data set.
COD_REALTIMESCROLL Chart FX will scroll the chart to the end of the data set.

Realtime Sample using the VBX-OCX

The following sample supposes that we have a timer that calls our application with two new values every second, so the idea is to inciluide thes points in a chart in RealTime mode:

```
Preparing the chart to be RealTime
This line of code can be avoided by including such constant in the chart_Create
function
Chart1.Type = Chart1.Type Or CT_EVENSPACING
Setting a buffer size of 50 points
Chart1.MaxValues = 50
Add a Loop marker and hiding the hourglass cursor
Chart1.RealTimeStyle = CRT_LOOPPOS | CRT_NOWAITARROW)
...
Finally when the timer calls set the new data
Chart1.OpenData(COD_VALUES Or COD_ADDPOINTS) = CHART_ML(1,2)
Set two new points of series 1
       Chart1.Value(0) = Rnd * 100
       Chart1.Value(1) = Rnd * 100
Close the channel forcing scroll
Chart1.CloseData(COD_VALUES Or COD_REALTIMESCROLL) =0
```

RealTime Charts. RealTime Style

The RealTime style refers to how the chart is going to be display in your application. Basically the two different settings available to this feature are showing the Loop Marker and hiding the hourglass cursor from the RealTime Chart.

Setting the RealTime style using the VBX-OCX

The property related in the VBX model is RealTimeStyle. This property is used to get/set the RealTime style of the chart. The prototype for this property is:

```
Visual Basic
[form.] Chart1.RealTimeStyle [ = setting$ ]
```

Setting can be a combination of:

CRT_LOOPPOS	Show Loop Marker
CRT_NOWAITARROW	Hide HourGlass cursor

To set both styles:

```
Chart1.RealTimeStyle = CRT_LOOPPOS Or CRT_NOWAITARROW
```

Real Time Charts. Loop Marker

The Loop Marker can be customized using the ItemColor, ItemWidth and ItemStyle properties using the **CI_LOOPPOS** index.

Real Time Charts. Scrolling Legends in Realtime

If youre working with a Realtime chart and also assigning legends to the points in the X axis, it is imperative that you scroll these legends in order to have your Realtime charts the appropiate legends everytime it receives new data.

Note: In the samples subdirectory you will find a Realtime sample that scroll the legends everytime the chart receives new information.

Scrolling Legends in RealTime using the VBX-OCX
In order to scroll the x axis legends in the VBX model you must follow these rules:

1) Include in your chart extended type the CTE_NOLEGINVALIDATE constant. Please refer to TypeEx property for more information.

2) Open the communication channel (OpenData property) in combination with the COD_ADDPOINTS constant (you must always do this when working with RealTime charts).

3) Set the value for the new point using the relative position in the Value property and have ready the legend that you want to assign to that new point.

4) If youre setting a BufferSize (Limited RealTime charts) with the MaxValues property you will have to erase the first legend everytime you receive a new point and the chart already completed its first cycle reaching the Max values limit.To erase the first legend you use the Legend property and set chr(1) to the Index 0.

5) After erasing the first legend you can set the new legend (also with the Legend property) including in the Index the MaxValue-1*** (for a 50 point buffersize you will set point legend no. 49, remember that all indexes in Chart FX are zero based), and the setting containing such legend.

6) Finally, in the CloseData property include the COD_SCROLLLEGENDS constant to force Chart FX to scroll the legends.
Note: scrolling legends does not apply when having the Loop marker on in your RealTime chart.

*****Important Note:** If you are not working with a buffersize assigned to your realtime chart (Max Values assigned) you must use the actual number of points in order to set the last point legend.You can obtain the number of points displayed in the chart with the NValues property, once you obtain the number of points (i.e. nPoints) you will use nPoints-1 index   instead of the MaxValue-1 index to set the legend.

Customizing Chart Painting. OverView

WARNING: This section explains a very advanced topic in Chart FX 3.0. In order to customize the chart painting you must be familiarized with the different available objects in the Windows environment (such as: pen, brushes, line styles, line, rectangles and circles primitives from the Windows API guide). You must be an advanced programmer to be able to customize chart painting appropiately. Therefore, if youre a novice windows developer or youre not familiarized in handling these objects in the Windows environment we suggest you continue working with the available API in Chart FX.

When customizing the chart paint process you, as a programmer, are able to capture three different events (notification messages) and different functions and properties that will allow you to place any object in the chart window (whether it is in the chart background or at top of the chart). These objects can be fonts, rectangles, circles, arrows, bitmpas and even propietary objects that you had created and know how to handle them appropiately (painting procedure) in any device context.
With this open architecture Chart FX provides virtually any kind of customization that you will need in your applications. Due to this fact, in this section we will describe the process of customizing the chart paiting with specific samples. Nevertheless, depending on your application you may want to use them differently. But remember that this a very advanced feature in Chart FX, and since you will be handling the Device Context is which Chart FX is painting the charts you may obtain erratic behavior if you paint different objects that you cant control.
You will receive three different notification messages as explained in the following diagram:

1) **PrePaint event** ------- 2) **PaintMarker event** ------- 3) **PostPaint event**

1) PrePaint event:
This event is sent before the chart is painted. Therefore, it is very useful for customizing the chart background. If you want place a gradient background or want to place a special picture or bitmap in the background chart this is the place to do it. Although the chart is not yet painted all the calculations for the markers and axis of the chart are available (CPI_*). please refer to following pages.

2) PaintMarker event:
This event is sent every time a marker is being painted. This event is very useful when you want to highlight certain information in your chart. You will be able to place any object you want highliting the marker that is being painted. Also in this event all the calculations for the markers and axis of the chart are available (CPI_*). Please refer to the following pages for more information.

3) PostPaint event:
After the chart finishes painting another event is posted for further customization. This event is very useful when you want to make final touches to the chart, like adding arrows, placing other fonts and general make-up to the final chart. Also in this event all calculations for the markers and axis (CPI_*) are still available. Please refer to following pages for more information.

Data

Customizing Chart Painting. PrePaint event

Please refer to RGB2DBk   or RGB3DBk properties to make the chart background transparent (CHART_TRANSPARENT) so the objects that you place in this code are visible when you finish the Painting process.

Important Note: for further information on the CPI_ constants used with the CM_GETPAINTFO message please refer to the following pages since the available CPI_ constants will allow you to retrieve very important information of where Chart FX is placing the different objects contained in the chart.

Placing a gradient background when processing the PrePaint the VBX model.

The following sample, as in all this manual for the VBX model, was created using Microsoft Visual Basic 3.0:

```
' Draw gradient background
    hDeviceC = Chart1.PaintInfo(CPI_GETDC)
    // get the chart position (usefull when printing or using chart_paint)
    lPos& = Chart1.PaintInfo(CPI_POSITION)
    x = CHART_LOWORD(lPos&)
    y = CHART_HIWORD(lPos&)
    hOldPen% = SelectObject(hDeviceC, GetStockObject(NULL_PEN))
    nHeight% = (h / 20) + 1
    nWidth% = (w / 20) + 1
    h = h + y
    w = w + x
    For i = 0 To 9
        l& = RGB(255 - (i * 20), 255 - (i * 20), 100)
        hBrush% = CreateSolidBrush(l&)
        hOldBrush% = SelectObject(hDeviceC, hBrush%)
        l& = Rectangle(hDeviceC, x + nWidth% * i, y + nHeight% * i, w - (nWidth% * i)
+ 1, h - (nHeight% * i) + 1)
        hOldBrush% = SelectObject(hDeviceC, hOldBrush%)
        hBrush% = DeleteObject(hBrush%)
    Next i
    hOldPen% = SelectObject(hDeviceC, hOldPen%)

    hDeviceC = Chart1.PaintInfo(CPI_RELEASEDC)
```

Customizing Chart Painting. PaintMarker event

As we mentioned, this event is very useful when you want to highlight or make-up the different markers (points, bars, etc) in the chart. This event is sent everytime a marker is going to paint, so you can retrieve important information (such as in waht position the marker is being painted) to customize the different markers in the chart.

In the following sample we are going to sorround with a rectangle only those points greater than fifty (50.00) in a line with point marker chart.

Note: This sample is also included in the custpain directory from Chart FX installation disk

Important Note: for further information on the CPI_ constants used with the CM_GETPAINTFO message please refer to the following pages since the available CPI_ constants will allow you to retrieve very important information of where Chart FX is placing the different objects contained in the chart.

Working with the PaintMarker event using   VBX-OCX

```
Chart1.ThisSerie = nSerie
    f# = Chart1.Value(nPoint)
    If f# > 50 Then
        nRadio% = 3 * Chart1.MarkerSize
        l& = nRadio%
        l& = chart_Send(Chart1.hWnd, CM_GETPAINTINFO, CPI_PRINTINFO, l&)
        If l& Then
            nRadio% = CHART_HIWORD(l&)
        End If
        hDeviceC = Chart1.PaintInfo(CPI_GETDC)
        hOldBrush% = SelectObject(hDeviceC, GetStockObject(NULL_BRUSH))
        i = Rectangle(hDeviceC, x - nRadio%, y - nRadio%, x + nRadio%, y + nRadio%)
        hOldBrush% = SelectObject(hDeviceC, hOldBrush%)
        hDeviceC = Chart1.PaintInfo(CPI_RELEASEDC)
    End If
```

Customizing Chart Painting. PostPaint event

Also in the list of customizing chart painting events is the PostPaint that will allow you to place objects in the chart window right after finishing the painting process for the whole charts. To show you some other applications of the chart painting process, the following sample will allow you to place a footer with the page number in a chart printout.

Important Note: for further information on the CPI_ constants used with the CM_GETPAINTFO message please refer to the following pages since the available CPI_ constants will allow you to retrieve very important information of where Chart FX is placing the different objects contained in the chart

Placing a footer with the page number using the CN_POSTPAINT event in the VBX model:

```
nPage% = Chart1.PaintInfo(CPI_PRINTINFO)
     it is printing ?
    If nPage% Then
        // get chart position
        lPos& = Chart1.PaintInfo(CPI_POSITION)
        hDeviceC = Chart1.PaintInfo(CPI_GETDC)
        s$ = "Page " + Str$(nPage%)
        i = TextOut(hDeviceC, CHART_LOWORD(lPos&) + w / 2, CHART_HIWORD(lPos&) + h,
s$, Len(s$))
        hDeviceC = Chart1.PaintInfo(CPI_RELEASEDC)
    End If
```

Customizing Chart Painting. Obtaining pertinent info.

When placing your objects you may want to know the location of the different items that are to be painted in the chart. For example, in order to highlight the point 4 in the chart you must know where this point is (coordinates) in order to be able to enclose it in a rectangle. The CM_GETPAINTINFO message with the appropiate CPI_* constant is used for this purpose.

Important Compatibility Issues:

We have implemented the GetPaintfo property to retrieve information from the painting process, but since properties only allow an Index this property (GetPaintInfo) is only useful when retrieving information with certain CPI_ constants, other CPI_ constants need another parameter in order to retrieve the appropiate information, in which you will need to fill the lParam of the   CM_GETPAINTINFO message under Visual Basic.
Also for those CPI_ constants that need a pointer in the lParam (feature not supported by Visual Basic) we have implemented the chart_GetPaintInfo function.
In each case we are explicitly describing which one should you use: GetPaintInfo property, CM_GETPAINTINFO message or chart_GetPaintInfo function.

In the following pages we will present the different CPI_ constants that are used with the CM_GETPAINTINFO message or as an Index of the GetPaintInfo property. But first lets take a look of the prototypes of each model:

CM_GETAPINTINFO message

This message is send through the chart_Send function to obtain pertinent information when customizing the chart painting:

Parameter	Description
wParam	CPI_ constant
lParam	Specific information about the CPI_ constant please refer to following pages

Comments

This message can also be used in the VBX-OCX model.

GetPaintInfo property

This property is used   to get pertinent information when customizing the chart painting. This property is read-only.

```
Visual Basic
[form.] Chart1.RealTimeStyle(Index)
```

Index is one the CPI_ constants (See following pages):
The CM_GETPAINTINFO message has to be used for certain CPI_ constants

CPI Constants.-

CPI_GETDC:

Description: Get the Device context of the chart so you can paint anything on it.
Important: If you are calling this info and you are not within paint events (PrePaint, PostPaint,PaintMarker) you must call the CPI_RELEASEDC when you finish.

lParam setting: NULL

Return Value: hDC
In the VBX use: GetPaintInfo property

CPI_ RELEASEDC:

Description: Release hDC when not used with any of the events (PrePaint, PaintMarker, PostPaint)
lParam setting: hDC returned in the CPI_GETDC
Return Value: None
In the VBX use: CM_GETPAINTINFO message with the chart_Send function

CPI_PIXELTOMARKER:

Description: Any given coordinate relative to the chart window is matched against its correspondent point-serie marker.
lParam setting: Long Value:
LOWORD=X
HIWORD=Y
Return Value: Long Value:LOWORD= nSerie
HIWORD= nPoint
nSerie is -1 if legend
nPoint is -1 if that location does not represent a point
In the VBX use: CM_GETPAINTINFO message with the chart_Send function

CPI_MARKERTOPIXEL:

Description: Transform the correspondent nSerie-nPoint to coordinates relative to the chart window.
lParam setting: Long Value:LOWORD=nSerieHIWORD=nPointSetting nSerie to -1 will obtain center.
Return Value: Long Value:LOWORD= XHIWORD=Y
In the VBX use: CM_GETPAINTINFO message with the chart_Send function

CPI_VALUETOPIXEL: Works with the Current. Axis See CurrentAxis property

Description: Receives a double value that is converted into its appropiate axis
lParam setting: Pointer to a double value
Return Value: Appropiate value.
In the VBX use: Use the chart_GetPaintInfo function.
Not applicable to rotated charts.

CPI_PIXELTOVALUE: Works with the Current. Axis See CurrentAxis property

Description: Receives a coordinate that is converted to a double value represented in the axis
lParam setting: Pointer to a double value
Return Value: NONE
But, the pointer is filled with the double value
In the VBX use: Use the chart_GetPaintInfo function
Not applicable to rotated charts.

CPI_POSITION

Description: Retrieves the upper-left corner of the chart
lParam setting: NONE
Return Value: Long Value:
LOWORD=left
HIWORD=top
In the VBX use: GetPaintInfo property

CPI_DIMENSION

Description: Retrieves the Width and Height of the chart (In pixels)

lParam setting: NONE

Return Value: Long Value:
LOWORD=Width
HIWORD=Height

In the VBX use: GetPaintInfo property

CPI_PRINTINFO

Description: This code is used when you want to know if the chart is being printed or you want to convert a value in printer coordinates.

lParam setting: NONE if you want to know if the chart is being printed OR
Value to be converted to printer coordinates.

Return Value: Long Value:
LOWORD=Page being printed, 0 = not being printed.
HIWORD=Converted value in printer coordinates

In the VBX use: GetPaintInfo property if you want to know only is the chart is being printed (lParam = NONE) OR CM_GETPAINTINFO message with the chart_Send function if you want to convert a number to printer coodrinates.

CPI_SCROLLINFO

Description: Retrieve the actual position and maximum number in the scroll of the chart

lParam setting: NONE

Return Value: Long Value:
LOWORD=Actual Pos.
HIWORD=Maximum

In the VBX use: GetPaintInfo property

CPI_3DINFO

Description: Retrieve the depth of each marker and total depth of the chart (Z axis dimension).

lParam setting: NONE

Return Value: Long Value:
LOWORD=Marker Depth
HIWORD=Total Depth (Z axis)

In the VBX use: GetPaintInfo property

CPI_3DTO2D

Description: Convert a coordinate from 3D to 2D.

lParam setting: Pointer to a CHART_P3D structure

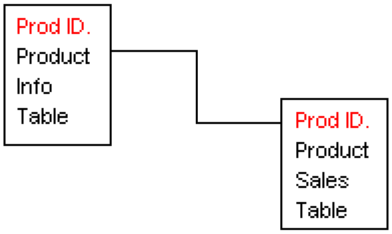
Return Value: Long Value:
LOWORD=Converted x
HIWORD=Converted y

In the VBX use: chart_GetPaintInfo function

DataBound

How can I bound Chart FX to a database?

Some development tools support DataBound capabilities, this feature will allow you to connect Chart FX Control (VBX or OCX only) to a database and retrieve the information directly from it. Instead of accessing the OpenData and CloseData to pass information (point by point)   to Chart FX you will pass a SELECT statement from the database and Chart FX will plot (and even assign legends!) automatically for you.
This feature is only available in the VBX and OCX models and in some development tools, please check your DataBound support in the development tool youre currently using to see if this feature is supported.
Chart FX is a multiple register control,which means that is going to take several register to plot them in the final chart.
Important: The following sample is included in the Chart FX sample sdirectory (for VB only) and this sample code guides you through the process of binding your chart control to an existing database.



In this sample we have a database (cfx30dat.mdb) with two tables: Product Information and Product Sales. Both of them are linked between the product ID. So for every product in the Product Information database there are one register for every month of the year.
We use two Data Controls (including 3 edit fields that show information stored in the database) in which you can scroll the information and the chart will change everytime you select another product from the Product Information database.
Important: We strongly suggest that you revised the sample included in your samples directory (realtime sub-directory in VB) so you can adapt this piece of code to your existing application.

Settings for Data Control #1 (Data1)
This control is bind to Product Information database, with the help of three edit fields, we show all the fields in the database (Including Product ID key field). In order to hook this control to the database, you have to:

1) Write on the DataBaseName property cfx30dat.mdb which is the database that contains both tables.

2) Write on RecordSource property the name of the Product Information Table (PRODUCTS). Please note that this RecordSource can be a sub-set of this table by placing a SELECT statement from that table.
Important Note: Since this control only contains a reference to the PRODUCTS table, this Data Control can be set at design time.

Settings for Data Control #2 (Data2)
Since the information in this data control and its edit fields depends on the settings of the Data1 Data control, we can not set the information at design time. Instead, we will use the Reposition event (of the

Data1 control) to fill out the information everytime the Data1 control changes its selection to another product. The code to place in the reposition event of the Data1 control will look like:

```
Sub Data1_Reposition ()
Data2.RecordSource = "Select sales,projected,date from PRODSALES where productid=" +
Text1.Text
Data2.Refresh
End Sub
```

2) Write on the DataBaseName property cfx30dat.mdb which is the database that contains both tables.

Linking the Chart Control to the Data Control

Finally, the Data2 control will contain the information we would like to plot (projected and sales fields). In order to accomplish this, the only set you have to make is link Chart FX object with the Data control. At design time you will write on the DataSource property of the chart the name of the data control you have the information in , in this case Data2.

Default Rules for DataBound charts.

Chart FX will apply default rules to construct the chart when linked to a Data control. These rules are somehow intelligent in picking the information from the database and assign the legends to it, so if you send a SELECT statement, Chart FX will create the chart series and point legends automatically. These rules are:
1) Series Legends will be taken from the numerical field names

2) All numerical columns will be plotted as different series and all string and/or date columns will be plotted as point legends (joined by the - character).

3) All string and numerical fields specified in the SELECT statement will be plot.

Changing the default Behavior of a Databound chart.

To change this default behavior Chart FX contains properties that will allow you to change this method of plotting the Values. Though these properties are only available at running time.

DataStyle Property = logical or of the following constants

CHART_DS_SERLEGEND = Take field names as series legends. Default = ON
CHART_DS_USEDATEASLEG = use date fields as legends. Default = OFF
CHART_DS_USETEXTASLEG = use text fields as legends. Default = ON

Note: When having different string or date fields Chart FX will construct a long string with every string and date field to assign to every legend point in the chart. If you want to avoid this behavior just turn OFF the appropiate constants using the DataStyle property.

DataType Property: Array property indicates the type of every field in the SELECT statement.

CDT_STRING = specify a string field type
CDT_NUMBER = specify a value field type
CDT_NOTUSED = do not use that field to plot in the chart.

Note: This property is very useful when you want to control how Chart FX retrieves and display the information from the database. For example in a 5 field SELECT statement such as:

```
Select year,sales,projected, returns, name from PRODSALES where prodid = 1234
```

The default behavior is that Chart FX will plot the year as another series since it is a number field and therefore it will be placed in the chart. Now, if the chart you want to make is one with the x axis containing the year and plot the sales and projected sales in a different series without using the return and name fields you will fill the DataType array as follows:

```
 1st we have to convert year field in a string to be selected as a x axis legend.
Chart1.DataType(0)=CDT_STRING
Then assigned the CDT_NUMBER constant to the number fields
Chart1.DataType(1) = CDT_NUMBER
Chart1.DataType(2) = CDT_NUMBER
Finally, assign CDT_NOTUSED to those fields we dont want to plot.
Chart1.DataType(3) = CDT_NOTUSED
Chart1.DataType(4) = CDT_NOTUSED
```

Creating Charts

You can create a chart in the same way you create any of the VBX-OCX objects in the development tool you're using:

Design Time: You just Draw the control and set the initial properties.
Run Time: Using the **Load** statement

At design time you can select the ChartType property or the Type property to control the type of chart you want to create (BAR, LINE, etc). Also important is to take a look at the Toolbar property to turn on the toolbar in the chart window.

Important note:
The chart_Create function in the VBX -OCX model can not be used.

Data

ChangeColor event

Sub *Chart1_ChangeColor (**nType** As Integer, **nIndex** As Integer, **nRes** As Integer)*

Description

This event is sent when any of the colors used in the chart is changed. This message is generated when the user drags any of the colors from the Palette Bar and drops it in any of the series of the chart.

Parameters	**Type**	**Description**
nType	Integer	CCC_SERIE, CCC_SERIEBK,CCC_ONE,CCC_ONEBK,CCC_BARHORZ, CCC_BKGND, CCC_2DBK, or CCC_3DBK.
nIndex	Integer	Color Index
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

Data

ChangeFont event

***Sub** Chart1_ChangeFont (**nIndex** As Integer, **nRes** As Integer)*

Description

This event is sent when any of the fonts used in the titles of the chart is changed. This message is generated when the user change the font used in any of titles of the chart by accesing the Edit Font Menu option or the toolbar button.

Parameters	**Type**	**Description**
nIndex	Integer	Font Index
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

Data

ChangePalette event

***Sub** Chart1_ChangePalette (**nIndex** As Integer, **nRes** As Integer)*

Description

This event is sent when any of the colors displayed in the Palette Bar is changed. This message is generated when the user double clicks any of the colors of the Palette Bar which accesses the colors commdlg to change that color of the Palette.

Parameters	**Type**	**Description**
nIndex	Integer	Palette Index
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

Data

ChangePattern event

***Sub** Chart1_ChangePattern (**nType** As Integer, **nIndex** As Integer, **nRes** As Integer)*

Description

This event is sent when any of the patterns used in the chart is changed. This message is generated when the user drags any of the patterns from the Palette Bar and drops it in any of the series of the chart.

Parameters	**Type**	**Description**
nType	Integer	CCP_SERIE or CCP_ONE.
nIndex	Integer	Pattern Index
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

Data

ChangePattPal event

***Sub** Chart1_ChangePattPal (**nIndex** As Integer, **nRes** As Integer)*

Description

This event is sent when any of the patterns displayed in the Pattern Bar is changed. This message is generated when the user double clicks any of the patterns of the Pattern Bar which accesses the pattern editor.

Parameters	Type	Description
nIndex	Integer	Pattern Index
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

Data

ChangeString event

***Sub** Chart1_ChangeString (**nType** As Integer, **nIndex** As Integer, **nRes** As Integer)*

Description

This message is sent when any of the text inside the chart is about to be change. The programmer may process this message and restrict (or allow) text changing in the chart. This message is generated when the user changes a string in the DataEditor nType As Integer = CCS_LEGEND or CCS_SERLEGEND.

Parameters	**Type**	**Description**
nIndex	Integer	Sring Index
nRes	Integer	0 Default processing
		1 Custom processing

Other Properties

None

Data

ChangeType event

***Sub** Chart1_ChangeType (**nType** As Integer, **nRes** As Integer)*

Description

This event is sent everytime the end users change a chart type using the menu of the toolbar.

Parameters	**Type**	**Description**
nType	Integer	New Chart Type
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

Data

ChangeValue event

***Sub** Chart1_ChangeValue (**dValue** As Double, **nSerie** As Integer, **nPoint** As Integer, **nRes** As Integer)*

Description

This event is sent when any of the values plotted in the chart is about to be change. The programmer may process this message and restrict (or allow) value changing in the chart. This message is generated when the user access the Data Editor and attempts to change any of the values displayed in the chart.

Parameters	Type	Description
dValue	Double	New Value
nSerie	Integer	Series Index
nPoint	Integer	Point Index
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

Data

GetLegend event

***Sub** Chart1_GetLegend (**bYLegend** As Integer, **nRes** As Integer)*

Description

This event is sent everytime Chart FX is going to paint a label in any of the axis supported. This event is very useful to customize legends. Please check the Customizing Y legends sample in the Programmers Guide section. Please refer to Other properties column for related messages.

Parameters	**Type**	**Description**
bYLegend	Integer	0 = X axis (Scatter) 1 = Y Axis 2 = Secondary Y Axis
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

In order to receive this event you must use the LegStyle property with the CL_GETLEGEND setting.With the HText property you can obtain the text that is about to be displayed in any of the axis

Data

GotFocus event

Sub *Chart1_GotFocus ()*

Description

This event is sent everytime the chart gains the focus in your application

Parameters

None

Other Properties

None

Data

InternalCommand event

***Sub** Chart1_InternalCommand (**wParam** As Integer, **lParam** As Long, **nRes** As Integer)*

Description

This event is sent everytime the end user presses any of the default buttons of the Toolbar.

Parameters	**Type**	**Description**
wParam	Integer	CFX_ID_ representing the pressed button.
lParam	Long	Not used
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

LButtonDblClk event

***Sub** Chart1_LButtonDblClk (**X** As Integer, **Y** As Integer, **nSerie** As Integer, **nPoint** As Integer, **nRes** As Integer)*

Description

This event is sent when the user makes a double-click with the left mouse button in any part of the chart. Normally, this message is used when the user double-clicks any point (Data Marker) in the chart in order to display a balloon help, a dialog, or a menu, or to call your own function that processes specific data.

Parameters	**Type**	**Description**
X	Integer	x coordinate
Y	Integer	y coordinate
nSerie	Integer	Series Inex
nPoint	Integer	Point Index
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

Data

LButtonDown event

***Sub** Chart1_LButtonDown (**X** As Integer, **Y** As Integer, **nRes** As Integer)*

Description

This event is sent when the user press down with the left mouse button in any part of the chart.

Parameters	**Type**	**Description**
X	Integer	x coordinate
Y	Integer	y coordinate
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

Use the CM_GETPAINTINFO message with CPI_PIXELTOMARKER constant to retrieve marker information of such location.

Data

LButtonUp event

***Sub** Chart1_LButtonUp (**X** As Integer, **Y** As Integer, **nRes** As Integer)*

Description

This event is sent when the user depress   the left mouse button after pressing it down.

Parameters	**Type**	**Description**
X	Integer	x coordinate
Y	Integer	y coordinate
nRes	Integer	0 Default processing
		1 Custom processing

Other Properties

Use the CM_GETPAINTINFO message with CPI_PIXELTOMARKER constant to retrieve marker information of such location.

Data

LostFocus event

Sub *Chart1_LostFocus ()*

Description

This event is sent everytime the chart losses the focus in your application.

Parameters

None

Other Properties

None

Data

Menu event

***Sub** Chart1_Menu (**wParam** As Integer, **nSerie** As Integer, **nPoint** As Integer, **nRes** As Integer)*

Description

This event is sent whenever the user presses the menu assigned to any of the button events. Please check the displaying a selection menu sample in the Programmers Guide section in this manual.

Parameters	**Type**	**Description**
wParam	Integer	ID of selected option
nSerie	Integer	Series Index
nPoint	Integer	Point Index
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

Data

MouseMove event

***Sub** Chart1_MouseMove (**X** As Integer, **Y** As Integer, **nRes** As Integer)*

Description

This message is sent when the user moves the mouse pointer over the chart window.

Parameters	**Type**	**Description**
X	Integer	x coordinate
Y	Integer	y coordinate
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

Use the CM_GETPAINTINFO message with CPI_PIXELTOMARKER constant to retrieve marker information of such location.

PostPaint event

***Sub** Chart1_PostPaint (**w** As Integer, **h** As Integer, **lPaint** As Long, **nRes** As Integer)*

Description
This event in send after the chart paint message starts and the calculations needed for painting the chart are made.

Parameters	**Type**	**Description**
w	Integer	width of rectangle in which the chart was painted
h	Integer	height of rectangle in which the chart was painted
lPaint	Long	Not used
nRes	Integer	0 Default processing 1 Custom processing

Other Properties
CM_GETPAINTINFO message will give you the device context in which the chart will paint. This message will return the HDC and other information related to the chart painting. Please Refer to Customizing Chart Painting topic.

PrePaint event

***Sub** Chart1_PrePaint (**w** As Integer, **h** As Integer, **lPaint** As Long, **nRes** As Integer)*

Description
This event in send before the chart paint message starts and the calculations needed for painting the chart are handy.:

Parameters	**Type**	**Description**
w	Integer	width of rectangle in which the chart is going to be painted
h	Integer	height of rectangle in which the chart is going to be painted
lPaint	Long	Not Used
nRes	Integer	0 Default processing 1 Custom processing

Other Properties
CM_GETPAINTINFO message will give you the device context in which the chart will paint. This message will return the HDC and other information related to the chart painting. Please Refer to Customizing Chart Painting topic.

Data

RbuttonDblClk event

***Sub** Chart1_RbuttonDblClk (**X** As Integer, **Y** As Integer, **nRes** As Integer)*

Description

This event is sent when the user makes a double-click with the right mouse button in any part of the chart

Parameters	**Type**	**Description**
X	Integer	x coordinate
Y	Integer	y coordinate
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

Use the CM_GETPAINTINFO message with CPI_PIXELTOMARKER constant to retrieve marker information of such location.

Data

RButtonDown event

***Sub** Chart1_RButtonDown (**X** As Integer, **Y** As Integer, **nSerie** As Integer, **nPoint** As Integer, **nRes** As Integer)*

Description

This event is sent when the user presses and holds the right mouse button. Normally, this message is used when the user clicks any point (Data Marker) in the chart in order to display a balloon help, a dialog, or a menu, or to call your own

Parameters	**Type**	**Description**
X	Integer	x coordinate
Y	Integer	y coordinate
nSerie	Integer	Series Index
nPoint	Integer	Point Index
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

RButtonUp event

***Sub** Chart1_RButtonUp (**X** As Integer, **Y** As Integer, **nRes** As Integer)*

Description

This message is sent when the user depress the rightmouse button after pressing it down.

Parameters	**Type**	**Description**
X	Integer	x coordinate
Y	Integer	y coordinate
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

Data

ReadFile event

Sub *Chart1_ReadFile ()*

Description

This message is sent after the user imports (or retrieves) a file that was previously saved. The programmer can not restrict this action, but can be notified that a new chart is being displayed. This message is generated when the user accesses the File Import menu option or the toolbar button

Parameters

None

Other Properties

None

Data

ReadTemplate event

Sub *Chart1_ReadTemplate ()*

Description

This message is sent after the user imports (or retrieves) a template that was previously saved. The programmer can not restrict this action, but can be notified that a new chart is being displayed. This message is generated when the user accesses the Template Import menu option.

Parameters

None

Other Properties

None

Data

ShowToolbar event

***Sub** Chart1_ShowToolbar (**nType** As Integer, **nRes** As Integer)*

Description

This event is sent everytime the Toolbar is show or Hide in the chart window. This event is used to customize the toolbar.

Parameters	**Type**	**Description**
nType	Integer	Not used
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

None

UserScroll event

***Sub** Chart1_UserScroll (**wScrollMsg** As Integer, **wScrollParam** As Integer, **nRes** As Integer)*

Description

This event is sent everytime the end users press any of the buttons or drag the thumb to a desired position to scroll between the points of the chart.

Parameters	**Type**	**Description**
wScrollMsg	Integer	Scroll Code (i.e. SB_LINEDOWN).
wScrollParam	Integer	Scroll Position
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

Please check SDK documentation for scroll codes and positions.

Data

Destroy event

***Sub** Chart1_Destroy ()*

Description

This event is sent every time the chart is destroyed. This event is very useul when is important to save or perform any customizing action when destroying the charts.

Parameters

None

Other Properties

None

Data

Paintmarker event

***Sub** Chart1_PaintMarker (**x** As Integer, **y** As Integer, **lPaint** As Long, **nSerie** As Integer, **nPoint** As Integer, **nRes** As Integer)*

Description

This event in sent every time a marker is being painted for customizing chart paiting.

Parameters	**Type**	**Description**
x	Integer	x coordinate in which the marker is being painted
y	Integer	y coordinate in which the marker is being painted
lPaint	Long	Not used
nSerie	Integer	Series Index of point that is being painted
nPoint	Integer	Point Index that is being painted
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

CM_GETPAINTINFO message will give you the device context in which the chart will paint. This message will return the HDC and other information related to the chart painting. Please Refer to Customizing Chart Painting topic.

Data

UserCommand event

***Sub** Chart1_UserCommand (**wParam** As Integer, **lParam** As Long, **nRes** As Integer)*

Description

This event is sent every time the user interacts with propietary items in the Toolbar. Propietary items are those items that you placed in the Toolbar when Customizing this tool. In order to receive an event for Toolbar default buttons please check InternalCommand event.

Parameters	**Type**	**Description**
wParam	Integer	Contains the ID assigned to such item
lParam	Long	Contains specific information of that item. (i.e. CBN_SELCHANGE for comboboxes)
nRes	Integer	0 Default processing 1 Custom processing

Other Properties

For additional information on how to place propietary buttons in the Toolbr, please refer to "Customizing the Toolbar" section in this electronic help file.

Technical Support

IMPORTANT NOTE: CHART FX Technical Support will not be provided through Software FX, Inc. Toll Free Number.

Technical Support is free and unlimited for 1-full year. Nevertheless, we will appreciate if you take the necesarry steps to assure that the information you need is not available in this manual or electronic help file. Since Chart FX is being revised and enhanced continously, we suggest that you check what version (or upgrade version) you are working with to make sure that your problem is not already fixed. We provide a WhatsNew.WRI file with all the updated information on the product.
Following is a list of possible items that may be appropiate to include in your posting:
0) Have your product serial number handy.
1) Provide the model of Chart FX 3.0 you're using: VBX, DLL or OCX.
2) Provide a discrete list of steps and conditions that fully reproduce the problem. If the problem is intermittent, describe the conditions, under which it arises in as much detail as possible.
3) List any error messages that appear on the screen.
4) Provide the name and version of the development tool you're currently using with Chart FX 3.0.
5) Provide a small code sample that demonstrates your use of Chart FX 3.0 or algorithm that is not working correctly. If you are having display problems please include a screen shot.
6) Provide as much detail about your system's hardware configuration as is relevant to the problem.

Please contact Software FX, Inc. at the following numbers:
Phone: (407) 998-2377
Fax: (407) 998-2383
CIS: 74032, 2412

Obsolete API from Chart FX 2.0

For those who started with Chart FX 2.0 we are including a list of all obsolete properties and messages that have been replaced or embbeded into others. Nevertheless, old messages and properties are still included and active in Chart FX 3.0.

WE STRONGLY SUGGEST THAT YOU CAREFULLY READ THE FOLLOWING TABLE AND MAKE THE APPROPIATE CHANGES TO YOUR EXISTING APPLICATION. ALTHOUGH ALL THE MESSAGES AND PROPERTIES FROM THE DLL AND VBX ARE STILL ACTIVE IN CHART FX 3.0 THEY WILL EVENTUALLY DISSAPPEAR.

Property	**Action (Description)**	**Replaced By or Embbeded In**
CopyBitmap	copy the chart to the clipboard (As a bitmap).	Export, ExportStr
CopyData	copy the data values of a chart to the Windows clipboard using a tab separated values format.	Export, ExporStr
ReadTemplate	retrieve and apply a previously saved template	Import, ImportStr
FixedBkColor	change the background color of the constant values passed to the chart.	ItemBkColor
FixedColor	change the color of the constant values passed to the chart.	ItemColor
FixedStyle	change the style of the lines used for the constant values passed to the chart	ItemStyle
FixedWidth	change the width of the lines used for the constant values passed to the chart.	ItemWidth
LegendWidth	change the width of legend window.	ToolSize
LineBkColor	change the background color of the lines for a 2D line chart.	ItemBkColor
LineColor	change the color of the lines for a 2D line chart.	ItemColor
LineStyle	change the style of the lines used in 2D Line Charts	ItemStyle
LineWidth	change the width of the lines used in 2D Line Charts	ItemWidth
PointType	change point type.	MultiPoint
CS_ constants	CS_ constants that specify a chart type (i.e CS_CHBAR) to allow the end user to change to an specific chart type from the end user menu	No longer available. Instead, you can use the GalleryTool property
CT_SCATTERLINE	To show connected lines in a scatter chart	include the CT_SHOWLINE constant instead. Also apply to POLAR charts.
CT_SHOWVALUES	To show values in markers in a line chart	Now applies to all chart types.
Hi-Low-Close	In this chart type (available as candlestick charts in 2.0)	Now you can include the CT_HILOWSTD constant in your type to display these charts in the standard mode.
WriteTemplate	save current attributes (colors, patterns, 3D View, rotation, etc) of a	Export, ExportStr

	chart so that they can be applied to other charts.	
CT_TOGETHER	To join bars or other markers in the chart. Specified in the chart Type	This constant was replace by the MarkerVolume property which controls the percentage of the marker
XLegType	To customize X axis legends	This property is replaced by LegStyle and has been enhanced to support other features
PrintIt	Print the chart in a full page mode	This message has been enhanced to specify a range of pages when printing in a full page mode.
Independent Styles in a 2D line chart.	In Chart FX 2.0 independent line style can not be set to the series.	MultiLineStyle is now available for this function.
Logarithmic scale	In Chart FX 2.0 logarithmic scale is not supported.	Adm Property has been enhanced to place a logarithmic scale to any of the axis. Also see: MultiYAxis

Data

Visual Basic

** Include the Chart FX VBX in your program*

- From the File menu select Add File... option.
- Choose chart2fx.vbx from your Windows directory.

** Include the Chart FX header file as follows.*

- From the File menu select Add File... option.
- Choose chart2fx.bas from c:\chartfx2\include.

Visual C++

* Include the Chart FX header file in your C++ code

```
       #include "c:\chartfx2\include\chart2fx.hpp"
```

* Enable the VBX engine at your application startup code.

- In the InitInstance member function add the following code:

```
              EnableVBX();
```

- To ensure proper response when your applcation cannot found the VBX file add the following code after the EnableVBX
Call

```
if (LoadVBXFile("CHART2FX.VBX") > HINSTANCE_ERROR)
             UnloadVBXFile("CHART2FX.VBX");
else {
AfxMessageBox("Cannot Load CHART2FX.VBX",MB_OK);
return FALSE;
}
```

* Include the Chart FX VBX button in the AppStudio's ToolBar.
    - From the File menu select Install Controls... option.
    - Choose chart2fx.vbx from the Windows directory.

Borland C/C++

** Include the Chart FX header file in your C++ code*

```
#include "c:\chartfx2\include\chart2fx.bch"
```

** Enable the VBX engine at your application startup code.*

- In the OWLMain function add the following code:

  TBIVbxLibrary vbxLib;

- In the application file (where OWLMain resides) add the following header file:

```
#include <owl\vbxctl.h>
```

** Include the Chart FX VBX button in the WorkShop's ToolBar.*

- From the File menu select Install Control Library... option.
- Choose chart2fx.vbx from the Windows directory.

SQL Windows

Gupta SQLWindows 4.0

** Include the Chart FX application library file in your program*

```
    * Application Description:
        * Libraries:
            * File Include: c:\chartfx2\include\chartfx2.apl
```

Gupta SQLWindows 4.1

** Include the Chart FX application library file in your program*

```
    * Application Description:
        * Libraries:
        * File Include: c:\chartfx2\include\chart2fx.apl
```

International Support

How can I translate Chart FX resources to my language?

Chart FX 3.0 installation disk provides an international support section that will allow you to access Chart FX resources (String Table, .RC, .DLG) to translate them to your own language and build a DLL (Dynamic Link Library) to be loaded dynamically at running time. In order to be able to build this DLL you should have a development tool that provides means of editing these resources (i.e. MS Visual C++, Windows SDK, MS AppStudio,. etc) and re-compile to build this new resource library:
In order to support all the posible foreign languages, we provide a special directory called INTSUP. In this directory CHART FX
install all the necessary files to make a DLL (Dynamic Link Library) with the resources that the library needs.

These files are:

File	Use	Proposed changes
IDMCHART.H	Header file	Do not change this file !
LANG30.DEF	Definition file	Change the LIBRARY topic to the name you will use for
LANG30.C	C Code	Do not change this file !
LANG30.H	Header file	Do not change this file !
LANG30.RC	Resource file	Change all the resources (Dialogs, Menu and String Table) to the language you need to support.

We also supply two makefiles to make even easier the process of making your own resources.

File	Development Tool
makefile	Microsoft C Compiler.
LANG30.MAK	Microsoft Visual C++ 1.51

Loading the new resources using the DLL
After editing the resources, recompile and build the new DLL you must use the Language property to load this library at running time. The following source code will load a new DLL called German.DLL

```
Chart1.Language  = German.DLL
```

License Agreement

Software FX, Inc. LICENSE CARD

THIS DOCUMENT IS VALUABLE PROPERTY. IT IS YOUR PROOF OF LICENSE FOR THE SOFTWARE FX, INC. PRODUCT YOU HAVE ACQUIRED AND YOU MUST RETAIN THIS SOFTWARE LICENSE CARD TO EXERCISE YOUR RIGHTS TO USE THE SOFTWARE.

READ CAREFULLY BEFORE OPENING SOFTWARE PACKET(S).

By opening the sealed packet(s) containing Software FX, Inc. software. (hereinafter "the Software" or "Software") , you are accepting the following License Agreement.

LICENSE AGREEMENT
This is a legal agreement between you (either an individual or an entity) and Software FX, Inc. By Opening the Sealed software packet(s) you are agreeing to be bound by the terms of this agreement. If you do not agree to the terms of this agreement, promptly return the unopened software packet(s) and the accompanying items (including written materials and binders or other containers) to the place you obtained them for a full refund.

SOFTWARE FX, INC. - SOFTWARE LICENSE
1. GRANT OF LICENSE. This License agreement permits you to use one copy of the enclosed software program (hereinafter "The SOFTWARE" or "SOFTWARE") on a single computer. The SOFTWARE is in "use" on a computer when it is loaded into temporary memory (i.e. RAM) or installed into permanent memory (e.g. hard disk, or other storage device) of that computer.

2. COPYRIGHT. The SOFTWARE is owned by Software FX, Inc. or its suppliers and is protected by United States copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE like any other copyrighted material (e.g. a book or a musical recording) except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes. or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You may not make multiples copies of SOFTWARE nor the written materials accompanying the SOFTWARE.

3. OTHER RESTRICTIONS. You may not rent or lease the SOFTWARE, but you may transfer the SOFTWARE and accompanying written materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement. Upon such transfer, you will notify Software FX, Inc. of the transfer and the name and address of recipient. You may not reverse engineer, decompile, or disassemble the SOFTWARE. If the SOFTWARE is an Update or has been updated, any transfer must include the most recent update and all prior versions.

4. DUAL-MEDIA SOFTWARE. If the SOFTWARE package contains both 3.5" and 5.25" disks, then you may use only the disks appropriate for your single-user computer. You may not use the other disks on another computer or loan, rent, lease, or transfer them to another user except as part of the permanent transfer (as provided above) of all SOFTWARE and written materials.

5. LIBRARY SOFTWARE. You have a royalty-free right to distribute only the "run-time modules " with the executable files created in any other vendor product (Language or Development Tool) limited as hereinafter set forth in paragraph a through d. Software FX, Inc. grants you a royalty-free distribution if : (a) you distribute the "run time" modules only in conjunction with the executable files that make use of them as a part of your software product; (b) you do not use the Software FX, Inc. name, logo or trademark to market your software product; (c) The SOFTWARE end users do not use the "run time" modules or any other SOFTWARE components for development purposes. and, (d) you agree to indemnify, hold harmless, and defend Software FX, Inc. and its suppliers from and against any and all claims or lawsuits including attorney's fees, that arise or result from the

use or distribution of your software product. If any of the conditions set forth in paragraphs a through d are breached, such breach shall constitute an unlawful use of the SOFTWARE, and you shall be prosecuted to the full extent of the law. Furthermore, you shall be liable to Software FX, Inc. for all damages caused by such breach and unlawful use of the software, including attorney's fees and costs incurred in any action, lawsuit or claim brought or filed to redress the breach of this agreement. The "run time modules" are those files included in the SOFTWARE package that are required during execution of your software program.

General Q&A Section

Data

Do all Chart FX models (VBX, DLL and OCX) provide the same functionality?

A: Yes! all models of Chart FX (VBX,OCX and DLL) provide the same functionality, the difference between them is how you interact with each model, using the DLL you will be using functions and messages. On the other hand, in VBX and OCX models you will interact with a chart by setting properties at design or running time. Even special editions of Chart FX (32-Bit DLL and OCX, 32-Bit IBM OS/2 2.1) provide the same functionality as the 16-bit editions (though being faster due to the 32-bit compatibility).

Data

What is the OCX model and why Chart FX is available to this technology?

A: OLE Controls (named OCX for their file extension) are custom controls based on Microsoft OLE 2.0 technology, which will standarize the use of custom controls for all available development tools. This technology is going to replace the current Visual Basic Controls (VBX) to provide a better interface of using controls and to provide porting capabilities to higher operating systems (i.e 32-Bit Chicago).OCXs will replace VBXs in upcoming versions of Microsoft Visual Basic and Microsoft Visual C++ and other development tools available from major vendors. According to this, Chart FX is ready to work with the new family of OLE 2.0 based development tools.

Data

Can I create scientific charts using Chart FX?

A: Depending on the scientific charts you want to make, although Chart FX is a business charting tool, we have implement a lot of features that can be used to make scientific charts. For example, Scatter Charts, 3D Surface Charts, Polar Charts and Spline Charts (or even a combination of Bar and Line!) can be performed without any problems in Chart FX to plot any kind of scientific data. Nevertheless, Chart FX is not able to perform XYZ Charts or linear regressions charts.

Data

How many chart types does Chart FX offer?

A: We have put a lot of effort in providing a great variety of types in Chart FX. Version 3.0 supports the following chart types:
LINE, BAR (Horizontal, Vertical and Gantt), SPLINE, MARK or POINTS, PIE, AREA, PARETO, SCATTER, HILOW (Open-Hi-Low-Close, Hi-Low-Close and Candlesticks charts), SURFACE, POLAR, CUBE, DOUGHNUT.
Also the ability to create Multiple Type Charts that contain different series in different types is also available in Chart FX 3.0.
A unique capability of Chart FX 3.0 is that ALL these charts support rotation and special 3D effects.

Data

Is there a limitation in the number of points that I can display in a chart?

A: Chart FX 3.0 (16 bit edition only) data segment is limited to 64K, which means a limitation in the number of points and series that you can send to a chart. Taking in mind this limit, you will be able to set up to 20000 points in total (i.e. 4 Series with 5000 points each). Please note that legends setting will also affect the data segment thus reducing the number of points and series that you can send to a chart. When using realtime charts we strongly suggest that you use Limited Real Time Charts since you can set a buffer size that will avoid data overflow in Chart FX. For more information on Limited Real Time Charts please refer to Real Time Support section in this manual.
Chart FX 32 bit editions (DLL or OCX) do not have a limitation in the number of points, series and legends that you can send to a chart.

Data

What is Smart Detection in Chart FX?

A: Chart FX supports event notification messages when the user interacts with the charts. This means that when the users make mouse clicks in the chart you, as a programmer, are able to capture that event and get information of the location in which the end user performed such action (either in screen coordinates or marker location).
This is a feature that is available in almost all charting libraries. But what happen when the charts have special 3D effects or rotation angles? -> Most of the charting tools are not able to notify you in which point and series the end users performed the click action. Chart FX support Smart Detection which will allow you to detect mouse events even when the charts have rotation angles and special 3D effect allowing you to have a better control of your application. This feature also allows the end users to be able to Drag & Drop Colors (using the PaletteBar or PatternBar).

Data

What End User Tools does Chart FX offer?

A: Chart FX is the only graphics library that allows you to display end users tools in the chart window, thus reducing your programming efforts to the minimum since all of the features that the library supports are available through these tools. Not only Chart FX provides standard tools embedded in the library but as well as having the capability to be customized to your needs. Chart FX provides the following end user tools:
Dockable & Customizable Toolbar: which will allow your end users to access the commands and functions with just a mouse click of the graphical buttons. Chart FX has a default toolbar that you can place anywhere in your chart window(even floating!) but if you need to place your buttons or controls (such as a combobox!) you can even customize this toolbar to meet your application interface and fucntionality.
We are also introducing the graphical icon combos that have an icon selection of colors and chart types.
Customizable PaletteBar: Which will allow your end users to select colors and Drag & Drop them to any part in your chart to customize the colors used in the chart, including series colors, background colors and legend colors. This tool is also customizable by performing a double-click and selecting the desired color from the Windows default palette bar.
Customizable PatternBar: Which will allow your end users to Drag & Drop patterns to the series of the chart and even change the default pattern by accesing the pattern editor built in chart FX.
Tabbed Dialog: Which will allow your end user access all the functionality of Chart FX in a very convenient way.
Data Editor: Which will allow your end users to access a grid embbeded in chart FX to edit and change the values that you initially sent to the chart.
Since Chart FX is a tool for developers you will have access to all end users tools and programatically set any of the features that are available to the end users. You, as a programmer, also decide what tools you want your end users to access and even restrict any portion of them.

Data

Is Chart FX available for 32-bit platforms development tools?

A: Yes! Chart FX have a special 32-Bit edition upgrade for the DLL (Dynamic Link Library) and OCX (Ole Control). Please contact Software FX, Inc. for pricing and availability of this special version of the library.

Data

What are Chart FX ToolTips?

A: Chart FX Toolbar supports customizable ToolTips, which are the descriptions of buttons. These can be customized to show a Balloon Help or a Word like rectangle containing the description. To customize them, just make a click with the right mouse button on the toolbar background and pick the desired selection.

Data

What enhancements were made to Chart FX 3.0?

A: Chart FX 3.0 is the result of our customers suggestions and needs. Therefore, you will find Chart FX 3.0 more suitable for specific tasks for your client application. We have put a lot of programming effort in extending our range for end user tools and chart aspects as well as introducing 4 new types of Charts (SURFACE, DOUGHNUT, CUBE and POLAR) for wider support. Also were introducing True Realtime Charting Capabilities and Multiple Type Charts to extend the range of supported features in the library. Chart FX 3.0 OLE Contros is also being introduced to support MS Access 2.0 as well as upcoming versions of Visual Basic and Visual C++.

If youre a Chart FX 2.0 user and want to know exactly the new feature set for Chart FX 3.0, please refer to Whats New in Chart FX 3.0 help file topic.

Font List

Constant	Description
CF_BOLD	Specifies whether the font is Bold
CF_ITALIC	Specifies whether the font is Italic
CF_UNDERLINE	Specifies whether the font is Underline
CF_STRIKEOUT	Specifies whether the font is Strikeout

/* Font Families Supported*/

Note: The font family is used in case the font specified is not found (or not installed) in Windows environment, by specifying this family Windows can create a similar font that you want to set.
To obtain more information about font families please refer to Windows SDK help file.

Constant	Description
CF_FDONTCARE	No family given
CF_FROMAN	Specifies whether the family is Roman
CF_FSWISS	Specifies whether the family is Swiss
CF_FMODERN	Specifies whether the family is Modern
CF_FSCRIPT	Specifies whether the family is Script
CF_FDECORATIVE	Specifies whether the family is Decorative

/* Font Typefaces Supported*/

Constant	Description
CF_ARIAL	Typeface is Arial
CF_COURIER	Typeface is Courier
CF_COURIERNEW	Typeface is Courier New
CF_HELVETICA	Typeface is Helvetica
CF_MODERN	Typeface is Modern
CF_ROMAN	Typeface is Roman
CF_SCRIPT	Typeface is Script
CF_SYMBOL	Typeface is Symbol
CF_TIMES	Typeface is Times
CF_TIMESNEWR	Typeface is Times New Roman
CF_WINGDINGS	Typeface isWingDings

Return Codes of Data Properties

Value	Meaning
CR_SUCCESS	Success.
CR_NOOPEN	OpenData property not called.
CR_OUTRANGE	Index used is greater than max number of items specified in OpenData
***CR_KEEPALL**	Success. Operation caused all previous values keeped.
***CR_LOSTLAST**	Success. Operation caused some values (last values) lost.

* This values can only be returned in a Value, IniValue or XValue properties.

CTBS_* Constants

Style	Hex Value	Description
CTBS_BUTTON	0x0001	Button
CTBS_MENU	0x0004	Menu Button. Must be used withCTBS_BUTTON
CTBS_2STATE	0x2000	2-State Button. Must be combined with CTBS_BUTTON
CTBS_REPEAT	0x0800	Timer Button. Must be used with CTBS_BUTTON.
CTBS_HEAD	0x4000	Identify the first button of a group.
CTBS_GROUP	0x1000	Identify the button is in the current group
CTBS_GROUP2STATE	0x3000	Combination of CTBS_2STATE and CTBS_GROUP.
CTBS_SEPARATOR	0x0002	Separator (Blank Space)
CTBS_ICONCOMBO	0x0008	Icon Combo (Only for Gallery Type and Color Palette).
CTBS_HWND	0x0088	User Controls
CTBS_DESTROY	0x0010	Combined with CTBS_HWND specify that Toolbar will destroy the control when the toolbar is show or hide.

Chart Types Table

This first set of constants are used to define the type of chart you want to change (Type Property), you should specify only one of them, which means that you will have undesired results if you combine them in a bitwise OR (i.e LINE | BAR). These are the basic type of charts included in the package. Nevertheless, you can add special effects (i.e. 3D, Rotation, Grid, etc.) depending on the type of chart you are working with.

Constant	Description
LINE	Line Chart
BAR	Bar Chart (Including Horizontal, and stacked charts)
SPLINE	Curve-fitting Chart
MARK	Point Chart
PIE	Pie Chart
AREA	Area Chart (Including stacked charts)
PARETO	Pareto Chart (Statistical Chart. Special)
SCATTER	Scatter Chart
HILOW	Hi-Low Close Chart
SURFACE	Surface Charts
POLAR	Polar Charts (also in 3D!)
CUBE	Cube Charts
DOUGHNUT	Doughnut Charts

The second set of constants are used to define several other aspects of the charts that can be very useful when you create the graph for the first time (instead of making several calls to other set functions). You combine them in a bitwise OR with the constants shown above. All of these are turned off by default, so you have to include them to activate these options.These are:

Constant **Description**

CT_3D To specify that the graph will be created or modified with 3D effect, if you include this constant the chart will be created as a 3D chart (if supported).

CT_HORZ This constant works only with Bar Charts, and if it is included the bar chart will be created as a horizontal bar chart.

CT_TOOL This constant specifies that the toolbar will be shown in the window containing the chart. This gives the end user access to the tools provided by the toolbar

CT_PALETTE This constant will turn on the Palette Bar, providing the end user the ability to change the colors of several objects in the chart, such as: series,
background, etc.

CT_PATTERN This constant will turn on the Pattern Bar, providing the end user the ability to change patterns used in the series of the chart.

CT_MENU This constant will turn on the menu of the chart, that provides the end user access to several options to modify the aspect of the chart.

CT_LEGEND This constant will turn on the value legend window in the chart. Default position = RIGHT

CT_SERLEGEND This constant will turn on the series legend window in a chart.

Default position = RIGHT

CT_POINTS This constant will show the points on a Line or Spline Chart.

CT_SHOWZERO This constant will cause a chart to set the starting point at zero. For example, if you have a bar chart with a minimum value of -50 and turn on this constant the starting point will be zero and you will have bars that go up or down, depending on their value.

CT_EACHBAR This constant is used to specify that a chart with a single series will have distinct colors at each data marker. i.e. Each bar will have different colors.

CT_CLUSTER This constant turn on the cluster options in which each data series is in its own row. To turn on this constant the CT_3D constant must be turned on.

CT_SHOWDATA This constant turns on the Data Editor (When this options is enabled the chart will not be visible).

CT_DLGGRAY This constant will cause the dialogs to be shown with a gray background, to provide support for applications that also use gray backgrounds. This keeps the graphics library consistent with the rest of the client application.

CT_COLORLINE This constant specifies that the lines of a 2D Line Chart must be drawn using colurs (the default behavior is to draw black lines)

CT_NOAREALINE This constant specifies that the vertical lines of an Area Chart will not be drawn.

CT_NOBORDERS This constant turns off the borders in bar charts.

CT_PIEVALUES This constant specifies that the values must be painted in the pie chart (instead of painting the percentages).

CT_SHOWLINES This constant specifies that lines will be shown between points in a Polar Chart.

CT_EVENSPACING This constant specifies points in the x axis will be even spaced, which means that points will be equally distanced in the x axis. When apply, this can cause a behavior in which you will see a blank gap at the right side of the chart, since the points can not be equally distance. Default = OFF

CT_PAINTMARKER This constant will turn on message event for customize chart drwaing process. Please refer to Customizing Chart Drawing topic in this manual for further information.

CT_SHOWVALUES This constant will display values above each marker inside the chart.

CT_HILOWSTD This constant is used to specify that the Hi-Low-Close charts will be display in a standard mode (Not Candlestick charts).

CT_TRACKMOUSE This constant have to be included for those chart types that you want to capture mouse tracking.

Chart Styles Table

With these constants you can restrict the access to several functions provided to the end user by the menu or toolbar. Include the following constants in a bitwise OR, and you will activate that feature for the end user.

Constant **Description**

CS_3D This will permit the end user to switch to 3D view any type of chart that is being displayed in 2D, this function can be accessed from the toolbar or from the menu.

CS_HORZ This will permit the end user to change the aspect of a bar chart to be displayed in horizontal bars. Since this type of chart belongs to the family of standard bar chart, you can restrict the end user from changing to a horizontal bar chart. This option can be accessed from the toolbar or the options dialog.

CS_SHOWPOINT This will permit the end user to show or hide the points in a line, spline or similar charts. This function can be accessed from the options dialog.

CS_SCALE This will permit the end user to change scale used on the Y-axis from the options dialog.

CS_TITLES This will permit the end user to change the text being assigned to the different titles supported (Top, Bottom, Right or Left). This function is provided in the options dialog.

CS_FONTS This will permit the end user to change the fonts used in any of the titles supported. This function is provided in the menu.

CS_EDITABLE This will permit the end user to change the values actually being graphed. This function is provided in the Data Editor.

CS_FILEEXPORT This will permit the end user to export (save) the current to a file. This function is provided in the toolbar or menu.

CS_FILEIMPORT This will permit the end user to import (open) a previously saved chart. This function is provided in the toolbar or menu.

CS_SCROLLABLE This will permit the end user to scroll if the current chart will not fit in the open window.

CS_PRINTABLE This will permit the end user to print the contents of the chart window. This function is provided in the toolbar and in the menu.

CS_3DVIEW This will permit the end user to modify the 3D View by accesing the 3D dialog where the user can rotate the view around the x or Y-axis.

CS_GRID This will permit the end user to modify the actual grids (Vertical, Horizontal or None) being displayed in the chart. This function is provided in the toolbar and in the menu.

CS_RESIZEABLE This will permit the end user to modify the internal borders of the chart. Note that if the chart is being displayed in a child window, the end user can resize the graph inside that window, but not the window itself.

CS_TEMPLATE This will permit the end user to operate (Save or apply) templates to a chart. This function is provided in the menu.

CS_COPY This will permit the end user to copy bitmap or data of actual chart to the clipboard, functionality provided in the toolbar and menu.

CS_MULTITYPE This will permit the end user to activate a MultiType chart from the Tabbed Dialog.

CS_CHDEFAULT This will permit the end user to access all the chart types available in Chart FX, from the Gallery Icon Combo provided in the Toolbar. Please refer to GalleryTool property for controlling access to different chart types

CS_CLOSEABLE When having an Overlapped chart window This will permit the end user to close the chart from the system menu..

CS_LOGSCALE This will permit the end user to switch from a Linear to Log scale in any of the axis from the Tabbed Dialog.

CS_ALL This will permit the end user to access all the functions explained above.

Basic. Creating a Simple Chart

How do I create a simple chart ?

You can create a chart in the same way you create any of the VBX-OCX objects in the development tool you're using:

Design Time: You just Draw the control and set the initial properties.
Run Time: Using the Load statement

At design time you can select the ChartType property or the Type property to control the type of chart you want to create (BAR, LINE, etc). Also important is to take a look at the Toolbar property to turn on the toolbar in the chart window.

Important note:
The chart_Create function in the VBX -OCX model can not be used.

Basic. Passing Data

How do I pass information (data) to a chart?

Once you have created the chart you need at least specify the data that you want to display, note that failing to do that will cause the library to show random values.
In order to specify the data to be shown, you must use the **Value** property, this is a single (float) property that must be used as an array property:

```
Chart1.Value(nPoint) = dValue!
```

This property tells the library that dValue is the value of the point nPoint in the serie "pointed" by the **ThisSerie** Property.

```
' Serie 0 , Point 3 , Value 10.5 */
Chart1.ThisSerie = 0
Chart1.Value(3) = 10.5
```

Nevertheless before using that property you need to be sure that the communications channel to the library is properly open. This is done through the pair of properties **OpenData** and **CloseData**.
Finally your code to set the data will look like this:

```
' Open the VALUES channel specifying 2 Series and 7 Points
Chart1.OpenData(COD_VALUES) = CHART_ML(2, 7)

' Code to set the data
Chart1.ThisSerie = 0
for i = 0 to 2 step 1
       Chart1.Value(i) = 9
next i
Chart1.ThisSerie = 1
for i = 0 to 2 step 1
       Chart1.Value(i) = 15
next i

' Close the VALUES channel
Chart1.CloseData(COD_VALUES) = 0
```

Basic. Scatter Charts

How do I create a scatter chart and pass information (data) to it?

Follow the steps explained in the previous sample of Creating a simple Chart, with the only excepton that youre going to specify SCATTER as your chart type (either at design time or at running time) and use the XValue property to assign the x values of each point in the chart. The source code should look as follows:

```
' Open both the VALUES and XVALUES channels
Chart1.OpenData(COD_VALUES) = CHART_ML(1, 7)
Chart1.OpenData(COD_XVALUES) = CHART_ML(1, 7)

' Code to set the data
Chart1.ThisSerie = 0
for i = 0 to 6 step 1
       Chart1.Value(i) = 9
       Chart1.XValue(i) = 6
next i
' Close both VALUES channels
Chart1.CloseData(COD_VALUES) = 0
Chart1.CloseData(COD_XVALUES) = 0
```

Tip 1: If you want lines to appear between the points in a scatter chart, just include the CT_SCATTERLINE constant using the Type property (either at design or running time).
Tip 2: You can set different series in the same scatter chart by increasing the number of series in the chart_OpenData function

Basic. Changing existing values

How do I change existing values in a chart or How do I add new information to an existent chart?

Once the chart is created, you can change any of the values displayed using the same properties explained in the chapter 1 Creating a simple chart: OpenData, Value, CloseData.
Setting the property OpenData with a new number of series and points will destroy existing data and prepare the communications channel to receive new data.

```
' Open the VALUES channel specifying 4 Series and 8 Points
' This call would destroy existent data
Chart1.OpenData(COD_VALUES) = CHART_ML(4, 8);
' Code to set the data
for i = 0 to 4 step 1
       Chart1.ThisSerie = i
       for j= 0 to 8 step 1
              Chart1.Value(i) = 12
       next j
next i
' Close the VALUES channel
Chart1.CloseData(COD_VALUES) = 0
```

If you only want to change the values without changing the number of points or series,   you can use the flag COD_UNCHANGE, which means that you will keep all the old data except for what you change with Value property.

```
' Open the VALUES channel and keep number of series and points
Chart1.OpenData(COD_VALUES) = COD_UNCHANGE
' Modify an arbitrary point
Chart1.ThisSerie = 1
Chart1.Value(4) = 10.5
' Close the VALUES channel
Chart1.CloseData(COD_VALUES) = 0
```

Tip 1: If youre changing existing values in a realtime mode (not adding new points but changing the existing ones), you may include the COD_SMOOTH constant in you CloseData property to avoid chart repaint flickering. Your CloseData will look like:

```
Chart1.CloseData(COD_VALUES Or COD_SMOOTH) = 0
```

Tip 2: If youre planning to add new points in a realtime mode and prevent chart flickering, please refer to Realtime Support.
Tip 3: You may want to check return values from the OpenData property when redimentioning your data arrays to see if you have loss previous data. These return codes are fully explained in the Passing data section in this help.
Tip 4: The end users are able to change the values that youve passed by accessing the Data Editor inside Chart FX, you can prevent them to do this by not including the CS_EDITABLE constant in your chart style (either at design or running time).

Basic. Passing Hidden Points

What are Hidden Points and how do I set them in a chart?

In some contexts of your application, your chart or some series in your chart, will not have all the points that you want to send. Eventually, these points can be middle points (i.e. point No. 3 and 6 of a 20 points chart) or ending points   (i.e. Point No. 7 and 8 of an 8 point chart). This situation can happen due to the lack of numbers in the database fields you want to plot or because you dont have these points available at chart creation time.

Anyway, you may want to set these points as hidden since they dont have an specific value in the chart. Even more if middle points are the missing ones, you dont want Chart FX to draw the lines between those hidden points in a line chart.

The way to specify hidden points in your chart is when you are sending the data values to it and those points missing or hidden should have the CHART_HIDDEN constant. The source code will look like:

```
' Open the VALUES channel specifying 4 Series and 8 Points
' This call would destroy existent data
Chart1.OpenData(COD_VALUES) = CHART_ML(4, 8);
' Code to set the data
for i = 0 to 4 step 1
       Chart1.ThisSerie = i
       for j= 0 to 8 step 1
              if (i=0 And j=6) Then
                     Chart1.Value(i)=CHART_HIDDEN
              Else
                     Chart1.Value(i) = 12
              End If
       next j
next i
' Close the VALUES channel
Chart1.CloseData(COD_VALUES) = 0
```

Although this code sets an arbitrary point as a hidden point you may want to detect which points are missing depending of the context of your application and set those who match the hidden pattern to the CHART_HIDDEN constant.

Basic. Rotate Charts Programatically

How can I rotate the chart without accessing the Rotation Dialog?

Although Chart FX provides the most intuitive way to rotate charts by accessing the rotation dialog which contains the axis and you or your end users drag marbles to desired rotation angles, you as a developer can chang the rotation dialog programatically or set the desired 3D angles through your own interface. This process is also useful when you want to animate charts by changing its rotation angles.
A very important thing in applying rotation angles is that you have to first turn on the 3D View option in order to be able to apply rotation angles to the chart.

In order to set rotation angles programatically you must first turn on the 3D View option with the View3D property and then assign the rotation agles using the Angles3D property. Your code will look like:

```
Set 3DVIEW to On
Chart1.View3D = TRUE
Set 45,45 rotation angles
Chart1.Angles3D = CHART_ML(45,45)
```

Tip 1: If you want to build your own rotation interface (i.e. using scroll bars hat set the rotation dialog around x and y axis you can use this message to apply he rotation dialog every time the user sets a position using this interface.

Basic. Change Default Series Colors

How can I change default series colors used by Chart FX?

When you create a chart, Chart FX selects default colors for your series, unless you send specific ones for the series in your chart. This default palette cycles every 16 colors, therefore if you have a chart with more than 16 series you will have to use this method of assigning colors to the different series in your chart.
The way to assign different colors is by opening a communications channel (OpenData) with the COD_COLORS constant and then set the desired RGB color to the series.

The following sample sets colors for a four series chart using the OpenData, Color and CloseData properties, assigning arbitrary RGB colors to each series.

```
Open the communication channel for 4 colors
Chart1.OpenData(COD_COLORS) = 4
Set the colors
Chart1.Color(0) = RGB(128,192,255)
Chart1.Color(1) = RGB(0,192,255)
Chart1.Color(2) = RGB(128,0,255)
Chart1.Color(3) = RGB(255,0,128)
Close channel
Chart1.CloseData(COD_COLORS) = 0
```

Basic. Save/Read Templates & Files

How can I save/read chart templates or chart files?

In Chart FX 3.0 you can save either the chart file (including data) or the chart template which will include the last configuration used (Colors, Patterns, 3D View, etc) without the chart data so you can apply the same aspects and characteristics to all of your charts. Saving a chart template will also affect the way the PaletterBar and PatternBar are displayed so if you want to change the default PaletteBar or PatternBar, edit the default colors (by double clicking it) and save a chart template with this settings and load it every time you create or load a chart.

With the Export property you can specify the type of file you want to save and the file (including path) in which you want to save the appropiate information. The source code should look like:

```
To save a chart file (including data)
Chart1.Export(CHART_CFXFILE) = c:\mychart.chf
 To save a chart template
Chart1.Export(CHART_CFXTEMPLATE) = c:\mytemp.ctm
```

With the Import property you can retrieve previosuly saved files and apply them to the actual chart. The source code will look like:

```
 To save a chart file (including data)
Chart1.Import(CHART_CFXFILE) = c:\mychart.chf
 To save a chart template
Chart1.Import(CHART_CFXTEMPLATE) = c:\mytemp.ctm
```

Important Information: The Export and Import message replace the ExportFile, ImportFile, WriteTemplate and ReadTemplate properties from Chart FX 2.0. Nevertheless, all those properties are still active in Chart FX 3.0 for Compatibility issues.

Basic. Setting Series/Points Legends

How do I set x axis and series legends?

In order to set points legends (x axis) you must use the Legend Property and to set the series legend you must use the SerLeg property. Normally, points legends that are to long will not fit in the x axis (this fact depends on the number of points youre setting to the chart. Therefore you can use the KeyLeg property to assign key legends to the x axis. When youve assigned key legends they will be placed in the x axis and in the points legend window will appear the text set with the Legend Property. Also, the LegStytle property is also useful to control several settings on how these legends are displayed in the x axis.

Important: When you set long text using the Legend Property and any of these labels do not fit in the x axis (due to the gap used between every point in the x axis) the default behavior of the library is to display the index of such point in red color to indicate that such text does not fit into that space. This behavior also apply when you dont set any legends to the x axis (red numbers indicating the index of the point). This behavior can be modified using the LegStyle property.
In the following sample we have a 5 point chart with two series. Every point represents the total sales in the first five months for two different products (A,B).

```
//Lets set the points legend
Chart1.Legend(0) = January
Chart1.Legend(1) = February
Chart1.Legend(2) = March
Chart1.Legend(3) = April
Chart1.Legend(4) = May
//Now the series legend
Chart1.SerLeg(0) = Product A
Chart1.SerLeg(1) = Product B
```

Tip 1: If you want the chart to initially display these legends when the chart is created just include the CT_LEGEND and CT_SERLEGEND in your chart type at design, or use the Type to set them programatically.
Tip 2: You can use the LegStyle property to control several settings of these legends.

Basic. Size/Separation of Markers

How do I control the size and separation of the markers in a chart?

When displaying a great number of points or when you have the need to control the separation between each point (x axis separation) Chart FX provides two methos of controlling the size and separation of the points in the x axis.

To control the size of the points in a line or similar chart you may use the MarkerSize property, which will allow you to change the size of the pioints for all the series in the chart. The MarkerVolume property is also provided to control the volume that each marker occupies in its corresponding x space, a very useful application of the MarkerVolume property is when you want your bars or cubes to appear joined in the chart (no blank gap at each side of the marker), you the apply a 100% volume which will cause the bars to occupy all the x space assign to it. Finally, the FixedGap property will allow you to control the space assigned to each point in the x axis, this property will allow you to place more points in on screen by assigning a small gap (in pixels) in the x axis.
In the following sample we will change the volume of a bar chart to joined them together and change the gap used in the x axis to fit more points in one screen:

```
The volume will affect to all series in the chart
Chart1.MarkerVolume = 100
Set fixedgap to 4 pixels to show more points
This property can also be set at design time
Chart1.FixedGap = 4
```

Tip 1: When changing the fixed gap using the FixedGap property to be a small number of pixels, if you had passed legends to the points of your chart using the Legend property, those legends will not be visible due to the lack of space in the x axis. You can control how you want Chart FX to behave in this case by setting the desired setting using the LegStyle property. Probably, you may want to hide the x axis labels, or make then vertical or even let them overwrite themselves.

Intermediate. Setting MultiType Charts

What is a Multiple Type Chart and how do I set it?

In Chart FX 3.0 you can create the most sophisticated MultiType charts without having to do special tricks or overlays between two different charts. These MultiType charts are those who have different series in different chart types (i.e. BAR, LINES, CUBES all mixed together in the same chart). You or your end users can specify which type apply to each series in your chart with just a message (DLL) or property set (VBX-OCX).
These charts also support special 3D effects, rotation capabilities and Smart Detection and can be combined with special Conic and Cilindric charts to add awesome charts to your application.
After your create your multiseries chart you can then assign the type you want for each series by doing the following:

With the MultiType property you can specify for a three series chart:

```
Chart1.MultiType(0) = AREA
Chart1.MultiType(1) = BAR
Chart1.MultiType(2) = CUBE
```

Also combined with the MultiType property you can add conic and cilindric shapes to BAR, CUBE or HILOW charts doing the following:

```
Set second series (BAR) to be cilindric base 5
Chart1.MultiShape(1) = 5
Set third series (CUBES) to be conic base 6
Chart1.MultiShape(2) = -6
```

MultiType charts can also be used to set different setting to different series in your chart. For example, if you want to show the values above each point only in the first series only of a three series line chart you will do the following:

```
Chart1.MultiType(0) = LINE Or CT_SHOWVALUES
```

Please refer to MultiType Property in this manual for more information on which settings (CT_) you can apply using this property.

Intermediate. Secondary Y Axis

How do I set secondary y axis and control both y axis settings?

Chart FX 3.0 supports 2 Y axis to assign different series to the main or secondary y axis. Also, any of these axis can be set to use a linear or logarithmic scale. You as a developer have access to customize the maximum, minimum, scale, gap and base used in any of these (This feature is also available to end users by accessing the scale section of Chart FX tabbed dialog). Since you can create a mutliseries chart with data that differs very much on the scale used , this feature will allow you to assign different series to a secondary y axis and your chart will maintain the same aspect in which you can see all the series.

Important: In your Chart FX installation directory you will see a sample that explains the process of having two different y axis and how to control settings for each one.

In order to create a secondary y axis in your chart you must use the MultiYAxis property, which will allow ou to create the secondary y axis and assign different series to this secondary axis. You have to make one call per each series that you want to assign to this secondary axis. Later, you can control the settings of this axis by accessing the Adm property. In the following piece of code we will set a secondary y axis with log scale while maintaining the main y axis linear:

```
Create a secondary y axis and assign second series
Chart1.MultiYAxis(1) = 1
Controling the settings of the secondary y axis
Assign a log base 10 scale to the scond y axis
Chart1.Adm(CSA_LOGBASE2) = 10
// Now were going to change the maximum used in primary y axis
Chart1.Adm(CSA_MAX) = 230
```

Tip 1: Since all the axis supported in Chart FX (x, main y and secondary y) can be set to use a log scale you must use the Adm property to create and assign the desired log scale to them. If you want to remove log scale and reconvert any of the axis to use a linear scale just pass zero (0) as the log scale specified as the setting of the Adm property when using any of the following constants: CSA_LOGSCALE, CSA_LOGSCALE2, CSA_LOGSCALEX.
Tip 2: Please refer to Adm property constants for controlling any of the axis settings.
Tip 3: Normally, when you change any of the existing values in a chart any of the y axis remain with the same minimum and maximum value. If you want Chart FX to automatically recalculate the minimum and maximum value everytime you change the existing values in a chart please refer to TypeEx property with the CTE_ACTMINMAX constant, which will turn on automatic recalculation of the axis

Intermediate. Customize Y Axis Labels

How can I customize y axis Legends?

Chart FX provides a notification message when placing legends in any of the axis. You can use this notification message to change the way labels are placed in any of the axis. For example, if you want to place a dollar sign in the y axis (or even format numbers with commas) you can perform this opeartion by getting the text that is about to be placed in the axis and modify it according to your needs. In the following sample we will place a dollar sign in front of the numbers displayed in the primary y axis.

In order to be able to capture the default text you must use the LegStyle property with the CL_GETLEGEND constant to make Chart FX notify you everytime is going to place a label in the y axis. After setting this message you will capture the GetLegend event to change the default text displayed in the y axis through the Htext property. The source code should look like:

```
Turn on the notification message
Chart1.LegStyle = CL_GETLEGEND
...
Later in the GetLegend Event you will place the following code:
If (bYLegend == 1) Then
Capture the default text
sLab = Chart1.HText
Format the string
sFinal = $ + sLab
Assign the string
Chart1.HText = sFinal
Stop default processing by assigning 1 to nRes parameter
nRes = 1
End If
```

Tip 1: In Visual Basic you may use the Format function to add commas to the numbers.
Tip 2: In the GetLegend Event you will receive two parameters: bYLegend which will contain and index specifying which axis is to be painted following these rules: 0 = X Axis, 1 = Primary Y Axis, 2 = Secondary Y Axis. The second parameter nRes must be set to 1 to stop default processing of placing these labels. **If you do not assign nRes to 1 Chart FX will proceed with the default behavior.**

Intermediate. Legend Positioning and Style

How can I control legend positioning and style ?

Chart FX provides ways to control location and style of all the tools provided in the library (Toolbar, Series Legends and points Legends), In the following sample we will control the legend positioning and style accordingly. You can also translate the following sample to also work with the toolbar. A very important issue is that points and series legends are separated so you can place them independently inside the chart window. Your end users are also able to control legend position and style by clicking with the righ mouse button to access Menus on demand with the different settings that they can apply to them.

In order to control the position of any of the legends (or toolbar) you must use the ToolPos property with the different pre-defined positions in Chart FX. Please remember that legends are considered tools inside the chart window and they have dockable options which will allow your end users to drag them to convert them to a floating tool and later on make them fixed anywhere inside the chart window. Also remember that you have to include the CT_LEGEND and/or CT_SERLEGEND constant in your type property to make any the appropiate legend visible. Also remember that you should set the appropiate legends before poisitioning or controlling the style for each one. Please refer to How do I set x and series legends also in this help.

```
Place the points legend at left side of the screen
Chart1.ToolPos(CTOOL_LEGEND) = CTP_LEFT
Make series legend floatable
Chart1.ToolPos(CTOOL_SERLEGEND) = CTP_FLOAT
```

Now with the ToolStyle property youre able to control the style for each legend in the chart (including toolbar). In the following we will make the series legend sizeable when child and points legends to accept double clicks to dock-undock

```
Make series legend sizeable with 3D frame
Chart1.ToolStyle(CTOOL_SERLEGEND) = CTS_SIZEABLE Or CTS_3DFRAME
 Make points legends to accept double clicks
Chart1.ToolStyle(CTOOL_LEGEND) = CTS_DBLCLKS
```

Please refer to ToolPos and ToolStyle properties for more information on legends positioning and styles.

Intermediate. Processing events

How can I process notification messages in Chart FX 3.0?

Notification messages (events) are the standard way child windows inform parent windows of changes and related information, this is the way all controls (ListBoxes, ComboBoxes, Edit Controls, etc.) work in Microsoft Windows and is also how CHART FX works.
The events supported in Chart FX allow you to capture end user actions in the chart so you can change default processing and give your application a special way to handle them. For example, capturing the double-click event in any marker of the chart will allow you to display any text within the default balloon help or even route your application to an specific module that handles such information, you can also capture special events to customize chart drawing and place your own objects in the chart.

All the events supported in the VBX-OCX are posted with their specific parameters. An important issue is for those events that you as a programmer can stop default processing. for such events we have included the nRes parameter that you have to set to 1 to force Chart FX to stop processing such event the usual way. For further information on how to handle notification messages please refer to Handling Notification messages in this help.
In this sample we are going to process the double-click event to show the balloon help if the value in which the end user makes the double click is between 25 and 50, otherwise we will disable the default balloon help.

```
Sub Chart1_LButtonDblClk (X As Integer, Y As Integer, nSerie As Integer, nPoint As
Integer, nRes As Integer)
Chart1.ThisSerie = nSerie
If (Chart1.Value(nPoint) > 25) And (Chart1.Value(nPoint) < 50) Then
    Chart1.HText = "Value Between 25 and 50"
Else
    Dont show the balloon help by stopping default processing
    nRes = 1
End If
End Sub
```

Intermediate. Displaying Internal Dialogs

If I dont show Chart FX Toolbar (or menu), can I popup Chart FX internal Dialogs to simulate toolbar behavior?

Depending on your application context you may want to control all end user interaction, which means that you dont want to show the Toolbar inside your chart window. If this is your case, you would probably be interested in showing any of the dialogs that Chart FX supports internally. For example, if you want to place a button in your application that controls the rotation angles, you may want to show the rotation dialog inside Chart FX.. We have implemented ways to call Chart FX internal dialogs which will allow you to control the access to those features inside your application.
In order to show any of the internal dialogs contained in Chart FX you must use the ShowDialog property with the appropiate index to the dialog you want to show. In the following sample we will show the Rotation Dialog:

```
// Show rotation dialog programatically
Chart1.ShowDialog(CDIALOG_ROTATE) = TRUE
```

Please refer to ShowDialog property for supported dialogs.

Intermediate. Customize X Axis Labels

How do I customize x axis labels?

In some cases, you may want to control how labels are placed in the x axis (as well as y and z axis). In Chart FX 3.0 you can control if the labels in the x axis are placed horizontally (default), vertical, staggered or even not show x axis labels at all.

With LegStyle property you are able to control label settings. In the following sample we will place x axis labels in a vertical mode and hide the y axis labels:

```
//Set Vertical X Legends
Chart1.LegStyle = CL_VERTXLEG
//Hide Y axis labels
Chart1.LegStyle = CL_HIDEYLEG
```

Please refer to LegStyle property for other settings.

Intermediate. Constant Lines & Color Stripes

What is the use for constant lines and color stripes and how can I set them?

In some cases, you may want to display one or several constant lines to highlight certain number in your chart, for example if youre plotting 1 series chart with 10 points and want to plot the average you may set this line without the effort and waste of adding a new series with the same amount of points and the same value. Instead, you can make Chart FX paint that horizontal line for you and you can even label it, change its color or style. On the other hand, if the case is that you want to represent a certain zone in your chart, for example an alarm zone between two different values (lets say from 180 to 200) you may want to use color stripes, which will allow you to highlight a certain range in your chart with an specific color.
The use for these two features are really helpful when you want to add specific effects to your charts.

The way to set these items in your chart, is by opening a communication channel (Opendata) and set the appropiate information to the chart. In the following sample we will create a constant line and two color stripes. Please note that you have to open two communication channels use the appropiate function (chart_SetConst for constants and chart_SetStripe for color stripes) and then close both communication channels. Were also customizing the label and the style of the constant line. Please note that constant and stripe passing muist be through functions instead of properties.

```
Open both communication channels (stripes and constants)
Chart1.OpenData(COD_CONSTANTS) = 1
Chart1.OpenData(COD_STRIPES) = 2
Set the constant at value 50
chart_SetConst(Chart1.hWnd,0,50)
Set two color stripes (20-40) and (80-100)
chart_SetStripe(Chart1.hWnd,0,20,40,RGB(128,255,0))
chart_SetStripe(Chart1.hWnd,1,80,100,RGB(255,0,0))
Close both channels
Chart1.CloseData(COD_CONSTANTS) = 0
Chart1.CloseData(COD_STRIPES) = 0
Now lets customize the label for the constant
Chart1.FixLeg(0) = Average
//Now lets customize the style for the constant
Chart1.ItemStyle(CI_FIXED) = CHART_DASHDOT
```

Tip 1: you can also use the ItemWidth property to set   the width of the constant line. Nevertheless, you will not be able to set a style for a constant line which width is bigger than 1 pixel.

Important: Note that in function calling you must first retrieve the chart window handle using the hWnd property.

Intermediate. Changing Balloon Text

How can I change default text in Chart FX balloon Help ?

When end users double-click in any particular point in the chart, Chart FX will display, by default, a balloon help containing the series legend, point legend and value of such point. You may capture this event (Please refer to Handling Notification Messages) and change this default text to any string you may want to show in that particular event.

You may capture the double-click event and even detect in which point the end user has made the double-click, this way you may check if you want to change the balloon help content using the HText. If you want to capture the default text that is to place in the balloon you may use the Htext property to retrieve such text. Lets suppose I want to capture the string and place the Hello! Im point: text before the default string:

```
Capturing default text
sDef = Chart1.HText
Modify default text
sFinal = Hello Im point No:  + sDef
Settingbthe text to be displayed
Chart1.HText = sFinal
```

Important Note: This piece of code have to be placed when capturing the LButtonDblclk event.

Intermediate. Line Styles in a 2D Line Chart

How do I set different line types and styles in a 2D Line Chart?

When printing or displaying a 2D line chart, if differentiating the series is an important matter you can assign colors to to every line independently. The problems arise when trying to print such chart (the colors are not distinguishable in the printout), therefore you may want to change styles and widths for every line in the chart independently. This will allow you to make a remarkable difference between every series in your chart.

Although we provide two properties related to line styles and widths (ItemStyle and ItemWidth) in a 2D line chart, they will apply to all the series in the chart. This means that using those properties and setting an specific style or width it will apply to all the series. Instead, you must use the MultiLineStyle property which will allow you to set styles and widths independently for every series in the chart.

Remember that you can set different colors to the lines (see Changing default colors used by Chart FX) and if you want to color the lines itself also remember that you have to include the CT_COLORLINE constant in your chart type at design time or set it programatically using the Type property.

In the following sample, we have a 3 series 2D line chart and were going to set a different style or width for each series using the MultiLineStyle property:

```
Chart1.MultiLineStyle(0) = CHART_ML(1,CHART_DASH)
Chart1.MultiLineStyle(1) = CHART_ML(3,CHART_SOLID)
Chart1.MultiLineStyle(2) = CHART_ML(1,CHART_DASHDOT)
```

Tip 1: If you want to clear all previosuly set styles or width set with MultiLineStyle property you can set Index to -1 and this will erase all style and width in just one step. Please refer to CM_MULTILINESTYLE message for more information on parameters setting.

```
Clearing all styles and widths
Chart1.MultiLineStyle(-1) = 0
```

Intermediate. Changing Type Programatically

How can I change (programatically) the chart type?

Controlling the Type of a chart programatically, can be very useful depending on the context of your application. Usually, the chart Type will allow you to change important characteristics of your chart. For example, you can show or hide any of the Tools programatically by setting an specific constant in your chart typem, or you can change from 3D mode to 2D mode, and other fetures contained in the CT_ type constants. The following samples will show you how to control type settings after you created the chart without altering the existing type.

The CT_ constants included in your header file handle different settings in your chart (not only the chart type itself). Usually, you will include all the types needed at design time by controlling two properties: ChartType which selects the chart type itself (BAR, LINE, etc) and the Type property, which will allow you set at design time all other CT_ constants needed in you chart. For example, to create a 3D Clustered bar chart with the toolbar included: Select BAR in your ChartType property and access the Type selection list and check the Clustered option in the list, to add the 3D effect just click on the Chart 3D property to switch between 2D and 3D mode.
In order to check the status of an specific bit inside the type word you can use the Type property in conjunction with the And operator, as follows:

```
Checking if the 3D bit is turned on
If (Chart1.Type And CT_3D)
       MsgBox 3D bit turned on
```

If for any reason you want to add or remove any chart type (CT_ constant) programatically, you may use the Type property at running time to turn on/off bits in the type word. In the following sample we will suppose that we have a button that switch to horizontal bar if the chart is a vertical bar chart and viceversa. For this sample we will work with the Xor operator

```
Swapping the CT_HORZ constant maintaining other styles
Chart1.Type = Chart1.Type Xor CT_HORZ
```

If you want to change the chart type itself (LINE,BAR, SPLINE, etc), we suggest you use the ChartType property instead of the Type property, since this will allow you to set only one the specific type constant. For example:

```
Chart1.ChartType = BAR
Chart1.ChartType = LINE
```

Tip 1: please refer to the Chart Types Table (CT_ constants) to get the list and descriptions for all the types supported in Chart FX.
Tip 2: The same samples apply to change the chart style, which controls the settings that are available to the end users in the chart.
Tip 3: Since the Type word is full (32-bit variable) we have included the TypeEX to set other types in your chart, the constants related to this message are those listed under the CTE_ prefix in your header file.

Advanced. Customizing Chart Painting

Please Jump to "Customizing Chart Painting" section of this help file.

Advanced. Customizing the Toolbar

Please Jump to "Customizing the Toolbar" Section of this help file

Advanced. RealTime Charts

Please Jump to "RealTime Charts" Section of this help file

Advanced. Capture Mouse to drag a point

How do I capture mouse events to allow my end users to drag a point to a desired location?

Chart FX now supports detection of all mouse events including MouseMove. This particular feature is very helpful when you want to provide your end users the ability to drag an specific marker to a desired location in the chart. In different contexts of your application your end users may want to interact with the chart to select an specific point and drag it over to another value.
The following sample code tracks mouse movements and change the cursor whenever the user reachs the top of a bar and allows them to drag the top of the the bar to control the height (value) of such marker.
The following is a transcript of the sample installed in Chart Fx samples directory.

The following sample code tracks mouse movements and change the cursor whenever the user reachs the top of a bar and allows them to drag the top of the the bar to control the height (value) of such marker.
The following is a transcript of the sample installed in Chart FX samples directory.

```
First you will need to declare the following
Declare Function LoadCursor Lib "USER.EXE" (ByVal hInst As Integer, ByVal nCursor As
Long) As Integer
Declare Function SetCursor Lib "USER.EXE" (ByVal hCursor As Integer) As Integer
Declare Function GetDC Lib "USER.EXE" (ByVal hWnd As Integer) As Integer
Declare Function ReleaseDC Lib "USER.EXE" (ByVal hWnd As Integer, ByVal hDC As
Integer) As Integer
Declare Function LineTo Lib "GDI.EXE" (ByVal hDC As Integer, ByVal x As Integer, ByVal
y As Integer) As Integer
Declare Function MoveTo Lib "GDI.EXE" (ByVal hDC As Integer, ByVal x As Integer, ByVal
y As Integer) As Integer
 Modify your chart Type to be able to track mouse events
Chart1.Type = (Chart1.Type Or CT_TRACKMOUSE)

Place the following code in the LButtonDown event
Sub Chart1_LButtonDown (x As Integer, y As Integer, nRes As Integer)
If bResize% And (nGlobalSeries% >= 0) And                      (nGlobalPoint
% >= 0) Then
  l& = CHART_ML(nGlobalSeries%, nGlobalPoint%)
  bDrag% = True
  nCursor% = LoadCursor(0, CHART_ML(32645, 0))
  n% = SetCursor(nCursor%)
  Chart1.MouseCapture = True
  nRes = 1
End If
End Sub
Place the following code in the lButtonUp event
Sub Chart1_LButtonUp (x As Integer, y As Integer,
                                  nRes As Integer)
If (bDrag%) Then
  bDrag% = False
  f# = y
  l& = chart_GetPaintInfo(Chart1.hWnd, 5, f#)
  Chart1.OpenData(COD_VALUES) =
              CHART_ML(COD_UNCHANGE, COD_UNCHANGE)
  Chart1.ThisSerie = nGlobalSeries%
  Chart1.Value(nGlobalPoint%) = f#
  Chart1.CloseData(COD_VALUES) = 0
  Chart1.MouseCapture = False
  nRes = 1
 End If

End Sub
```

Place the following code in the MouseMove event

```
Sub Chart1_MouseMove (x As Integer, y As Integer,
                                 nRes As Integer)
If (Not bDrag%) Then
   lPos& = CHART_ML(x, y)
   lMarker& = chart_Send(Chart1.hWnd, CM_GETPAINTINFO,
       CPI_PIXELTOMARKER, lPos&)
   nSerie% = CHART_LOWORD(lMarker&)
   nPoint% = CHART_HIWORD(lMarker&)
   Text1.Text = nSerie% + 1
   Text2.Text = nPoint% + 1
   If (nSerie% >= 0) And (nPoint% >= 0) Then
   ' Capture the top position of the marker (top edge)
   lPos& = chart_Send(Chart1.hWnd, CM_GETPAINTINFO,
       CPI_MARKERTOPIXEL, lMarker&)
   XP% = CHART_LOWORD(lPos&)
   YP% = CHART_HIWORD(lPos&)
   If (y >= YP% - 2 And y <= YP% + 2) Then
      nCursor% = LoadCursor(0, CHART_ML(32645, 0))
      bResize% = True
      nGlobalSeries% = nSerie%
      nGlobalPoint% = nPoint%
      lGlobalPos& = lPos&
    Else
       bResize% = False
       nCursor% = LoadCursor(0, CHART_ML(32515, 0))
    End If
    n% = SetCursor(nCursor%)
    nRes = 1
     End If
End If
If bResize% And bDrag% Then
   nCursor% = LoadCursor(0, CHART_ML(32645, 0))
   n% = SetCursor(nCursor%)
    nRes = 1
End If
End Sub
```

Advanced. Gradient Background

How do I place a gradient background in my chart?

One of the applications of customizing chart drawing process is the ability to place a gradient in the chart background. This feature will allow you to have awesome presentation graphics and even develop an interface so your end users may select the colors they desired as the gradient in the background.

Important:The following code is a transcript from the sample located in the custpain sub-directory

The trick in making a gradient on the chart background is to make the background of the chart itself (3D or 2D wall) transparent, so this gradient might be seen in al chart extension, after making the chart background transparent, process the PrePaint event and draw the gradient underneath the chart:

```
' Transparent backgrounds
   Chart1.RGB3DBk = CHART_TRANSPARENT
   Chart1.RGB2DBk = CHART_TRANSPARENT
Sub Chart1_PrePaint (w As Integer, h As Integer,
                      lPaint As Long, nRes As Integer)
    ' Draw gradient background
    hDeviceC = Chart1.PaintInfo(CPI_GETDC)
    lPos& = Chart1.PaintInfo(CPI_POSITION)
    x = CHART_LOWORD(lPos&)
    y = CHART_HIWORD(lPos&)
    hOldPen% = SelectObject(hDeviceC,GetStockObject(NULL_PEN))
    nHeight% = (h / 20) + 1
    nWidth% = (w / 20) + 1
    h = h + y
    w = w + x
    For i = 0 To 9
       l& = RGB(255 - (i * 20), 255 - (i * 20), 100)
       hBrush% = CreateSolidBrush(l&)
       hOldBrush% = SelectObject(hDeviceC, hBrush%)
       l& = Rectangle(hDeviceC, x + nWidth% * i, y + nHeight% * i, w - (nWidth% * i) + 1, h - \
        (nHeight% * i) + 1)
       hOldBrush% = SelectObject(hDeviceC, hOldBrush%)
       hBrush% = DeleteObject(hBrush%)
    Next i
    hOldPen% = SelectObject(hDeviceC, hOldPen%)
    hDeviceC = Chart1.PaintInfo(CPI_RELEASEDC)
End Sub
```

Advanced. Print Several Charts

How can I print several charts in the same page or a chart with other objects in my printout?

Chart FX supports printing in two ways. You can use the CM_PRINT message (DLL model) or the PrintIt property (VBX-OCX model) to print the chart in a full page mode (including margins specified in the Page Setup dialog) and taking the default paper orientation set in the Printer Setup of Windows. This feature is also available to the end users by pressing the print button in the Toolbar.
In some cases you will want to print several charts in the same page or even print a chart with some other objects in your printout. For this purpose Chart FX provides the chart_Paint function which will allow you to pass a Device Context (and dimensions within it) in which the library will paint the chart in. If you want to be able to customize the printing process you can pass the Printer Device Context and dimensions in which Chart FX is to print the chart.

Important: A full sample of printing two charts in the same page has been installed in your samples directory.

The following sample prints two different charts in the same page using the chart_Paint function:

```
Sub PrintFillingPage ()
    Dim l, r, t, b As Integer

    Printer.Print ""
    px = Printer.TwipsPerPixelX
    py = Printer.TwipsPerPixelY
    w = Printer.Width
    h = Printer.Height

    gap = 100 / px
    t = gap
    b = ((h / 2) / py) - gap

    l = gap
    r = (w / px) - gap / 2
    Call chart_Paint(Chart1.hWnd, Printer.hDC, l, t, r,     b, True, 0)
    t = b
    b = (h / py) - gap
    Call chart_Paint(Chart2.hWnd, Printer.hDC, l, t, r,     b, True, 0)
    Printer.EndDoc
End Sub
```

Advanced. Creating a Status Bar

How do I create a status bar in my chart Window?

Chart FX has built-in code to create and draw StatusBars, this kind of window is widely used in many comercial applications to inform the end user of the status of the program and relevant information. This tool is intended to display information, this means that your end users will not be able to interact with it.

If the development tool you are using does not provide user-defined structures (or casting) the easiest way to create a status bar is using OpenData and CloseData properties in conjunction with the chart_SetStatusItem function

```
chart_SetStatusItem(HWND hwndChart,int nItem,BOOL bText, UINT wIdm,...)
```

The parameters of the chart_SetStatusItem are the window handle of the chart, the number of items to set and the rest of the parameters (excepting bText) are the same as explained in the CHART_STITEM structure. The only limitation is that you cannot initialize text items.

An example code to create a StatusBar would be:

```
' Open the communications channel
Chart1.OpenData(COD_STATUSITEMS) = NUMITEMS

' Set the items
hWnd = Chart1.hWnd
chart_SetStatusItem(hWnd,0,TRUE,IDM_TEXT1,TRUE,100,50,4,CHART_STLEFT)
chart_SetStatusItem(hWnd,1,TRUE,IDM_TEXT2,TRUE,80,80,5,CHART_STCENTER)
chart_SetStatusItem(hWnd,2,FALSE,NULL,TRUE,40,40,10,NULL)
chart_SetStatusItem(hWnd,3,TRUE,IDM_TEXT3,TRUE,50,30,2,CHART_STRIGHT)
' Close the items channel
Chart1.CloseData(COD_STATUSITEMS) = 0
```

The code needed to show a StatusBar is:

```
' TRUE means Show, FALSE means HIDE
Chart1.ShowStatus = TRUE
```

The code needed to modify a StatusBar text item is:

```
' wIdm is the UINT code you assign to the item at creation time
Chart1.StatusText(wIdm) = "New Text"
```

Advanced. Display Menu when Right click

How can I customize Right Button click to display a selection menu?

When capturing the double-click or right click event you can display a Menu on demand in your chart. This feature will allow you to integrate your application with the charting module in a very convenient way. For example, if you have an specific module in your application that retrieves the history data for certain point in the chart, you can display a menu when the end user makes the click over any marker in the chart, select the option and then process this message to re-route your application to the correct module. Anyway the following will show you how to display a menu when the end user makes a right click of the mouse.

If you need to provide a menu for the right click of the mouse you have to keep four things in mind:
a. How to tell CHART FX to use a menu

```
Chart1.RigClk(CHART_MENUCLK) = hMenu
```

b. How to handle messages generated by the menu.
CHART FX will send you a Menu Event.

c. When to create and destroy the menu:
Note that you can not destroy the menu once you use RigClk (DblClk) property because CHART FX use the handle you provided, so we recommend that you create the menu when your application start and destroy the menu when your application finish.
If the chart is a child window you can create the menu in the Load event of the parent window and destroy it in the Unload event.

d. How to create and destroy the menu:
CHART FX will use the TrackPopupMenu function so your menu will have to be a Popup menu, the windows functions that you need are CreatePopupMenu or GetSubMenu and DestroyMenu.

Utilities

About Property

✓ Design Time ✗ Run Time

The About property will allow you to retrieve important information about Chart FX development team.

Type/Style

Type Property

✓ Design Time ✓ Run Time

A long value (32 bits) that sets or returns the type of the chart, this type includes gallery type as well as other visual elements in the chart window. The default value is LINE | CT_SHOWPOINTS.

Visual Basic

```
[form.] Chart1.Type [ = setting& ]
```

Visual C++

```
lType = pChart1->GetNumProperty("Type");
pChart1->SetNumProperty("Type",lSetting);
```

SQLWindows

```
Set lType = chart_GetNumProp(cc1,"Type")
Call chart_SetNumProp(cc1,"Type",lSetting)
```

Borland C++

```
pChart1->GetPropType(lType);
pChart1->SetPropType(lSetting);
```

Property Code

CP_TYPE

Remarks

Setting of this property must contain a bitwise OR of Chart Types constants

Data Type

Long

See Also

TypeEx, Style, StyleEX, Chart Types Table

Style Property

A long value (32 bits) that sets or returns the style of the chart. This style refers to what the end user can do in the chart window, thus permitting to change from one type of chart to another, modify 3D View, Rotation, etc.

Visual Basic

```
[form.] Chart1.Style [ = setting& ]
```

Visual C++

```
lStyle = pChart1->GetNumProperty("Style");
pChart1->SetNumProperty("Style",lSetting);
```

SQLWindows

```
Set lStyle = chart_GetNumProp(cc1,"Style")
Call chart_SetNumProp(cc1,"Style",lSetting)
```

Borland C++

```
pChart1->GetPropStyle(lStyle);
pChart1->SetPropStyle(lSetting);
```

Property Code

CP_STYLE

Remarks

Setting of this property must contain a bitwise OR of Chart Styles constants

Comments

CHART_ADD	Add selected styles to chart.
CHART_REMOVE	Remove style from chart.
CHART_SET	Remove previous styles and set styles specified in lParam.

Data Type

Long

See Also

StyleEX, Type,TypeEx, Chart Styles Table

NSeries Property

This property will allow you to retrieve the number of points or series in the chart. When set at design time, this property will assign random data to the chart. At runtime this property is read-only.

Visual Basic
```
[form.] Chart1.NSeries
```

Visual C++
```
n = pChart1->GetNumProperty("NSeries");
```

SQLWindows
```
Call chart_GetArrNumProp(n,"NSeries")
```

Borland C++
```
pChart1->GetPropNSeries();
```

Property Code
CP_NSERIES

Data Type
Integer

See Also
NValues, Passing Data to Chart FX

NValues Property

This property will allow you to retrieve the number of points or series in the chart. When set at design time, this property will assign random data to the chart. At runtime this property is read-only.

Visual Basic
```
[form.] Chart1.NValues
```

Visual C++
```
n = pChart1->GetNumProperty("NValues");
```

SQLWindows
```
Call chart_GetArrNumProp(n,"NValues")
```

Borland C++
```
pChart1->GetPropNValues();
```

Property Code
CP_NVALUES

Data Type
Integer

See Also
NSeries, Passing Data to Chart FX

OpenData Property

✘ Design Time ✓ Run Time

Setting this property open a communication channel to send data to the chart object, the index represents the type of channel to be opened and the value represents the number of items.

Visual Basic
```
[form.] Chart1.OpenData(Index) = setting&
```

Visual C++
```
pChart1->SetNumProperty("OpenData",lSetting,Index);
```

SQLWindows
```
Call chart_SetArrNumProp(cc1,"OpenData",lSetting,Index)
```

Borland C++
```
pChart1->SetPropOpenData(lSetting,Index);
```

Remarks

Once the data is filled, you must use CloseData property in order to close the communication channel. The indexes that can be used with this property and the meaning of these settings are:

Index	What the value represents	Related Property
COD_VALUES	CHART_ML(nSeries,nPoints)	Value
COD_CONSTANTS	nConstants	Const
COD_COLORS	nColors	Color
COD_STRIPES	nStripes	chart_SetStripe
COD_INIVALUES	MAKELONG(nSeries,nPoints)	XValue
COD_XVALUES	MAKELONG(nSeries,nPoints)	IniValue
COD_STATUSITEMS	nStatusItems	chart_SetStatusItem

OpenData can also be used with a combination of:

COD_ADDPOINTS	RealTime Usage. Relative pointer access
COD_RESETMINMAX	Recalculate Min-Max when setting new values. Use this constant only when passing a whole new set of values

See Also

CloseData, Passing Data to Chart FX, Value, XValue, IniValue, Color

CloseData Property

By setting this property you close the communications channel opened with the OpenData Property, It's extremely important that you close all the opened channels.
The value assigned to this property is not used (must be set to zero).

Visual Basic
```
[form.] Chart1.CloseData(Index)  =  0
```

Visual C++
```
pChart1->SetNumProperty("CloseData",0,Index);
```

SQLWindows
```
Call chart_SetArrNumProp(cc1,"CloseData",0,Index)
```

Borland C++
```
pChart1->SetPropCloseData(0,Index);
```

Property Code
CP_CLOSEDATA

Remarks
The following COD_ constants can be used when closing a coomunication channel. Combined with a logical OR containing the appropiate COD_ constant specified in the OpenData function:

COD_SCROLLLEGEND	Scroll Legends in RealTime charts
COD_NOINVALIDATE	Do not invalidate chart for repainting
COD_SMOOTH	Apply a BitBlitz technique when repainting. This will cause the chart to repaint without flickering.
COD_REALTIMESCROLL	For RealTime charts. Scroll to the last acquired point.
COD_REALTIME	For RealTime charts. Do not scroll.
COD_REMOVE	Used in RealTime charts. To delete all data related to the appropiate COD_ constant (i.e. COD_VALUES).

Data Type
Long

See Also
OpenData, Passing Data to Chart FX

Data

Value Property

Data

A float property (Single Type in Visual Basic) that sets or returns the value of a point in the chart, the point to set (get) is represented by the index and the serie is given by the ThisSerie Property.

Visual Basic

```
[form.] Chart1.Value(Index) [ = setting! ]
```

Visual C++

```
fValue = pChart1->GetFloatProperty("Value",Index);
pChart1->SetFloatProperty("Value",fSetting,Index);
```

SQLWindows

```
Set fValue = chart_GetNumProp(cc1,"Value",Index)
Call chart_SetNumProp(cc1,"Value",fSetting,Index)
```

Borland C++

```
pChart1->GetPropValue(fValue,Index);
pChart1->SetPropValue(fSetting,Index);
```

Property Code
CP_VALUE

Remarks
Before using this property the COD_VALUES communication channel must be opened with the OpenData Property.

Data Type
Float (Single)

See Also
OpenData, CloseData, Passing Data to Chart FX, XValue, IniValue

XValue Property

A float property (Single Type in Visual Basic) that sets or returns the X-axis value of a point in the chart, the point to set (get) is represented by the index and the serie is given by the ThisSerie Property.

Visual Basic

```
[form.] Chart1.XValue(Index) [ = setting! ]
```

Visual C++

```
fXValue = pChart1->GetFloatProperty("XValue",Index);
pChart1->SetFloatProperty("XValue",fSetting,Index);
```

SQLWindows

```
Set fXValue = chart_GetArrNumProp(cc1,"XValue",Index)
Call chart_SetArrNumProp(cc1,"XValue",nSetting,Index)
```

Borland C++

```
pChart1->GetPropXValue(fXValue,Index);
pChart1->SetPropXValue(fSetting,Index);
```

Property Code
CP_XVALUE

Remarks
Before using this property the COD_XVALUES communication channel must be opened with the OpenData Property. This property is supported by scatter charts only.

Data Type
Float (Single)

See Also
OpenData, CloseData, Passing Data to Chart FX, Value, IniValue

Data

IniValue Property

Data

A float property (Single Type in Visual Basic) that sets or returns the initial value of a point in the chart, the point to set/get is represented by the index and the series is given by the current value of the ThisSerie Property.

Visual Basic
```
[form.] Chart1.IniValue(Index) [ = setting! ]
```

Visual C++
```
fIniValue = pChart1-> GetFloatProperty("IniValue",Index);
pChart1->SetFloatProperty("IniValue",fSetting,Index);
```

SQLWindows
```
Set fIniValue = chart_GetArrNumProp(cc1,"IniValue",Index)
Call chart_SetArrNumProp(cc1,"IniValue",fSetting,Index)
```

Borland C++
```
pChart1->GetPropIniValue(fIniValue,Index);
pChart1->SetPropIniValue(fSetting,Index);
```

Property Code
CP_INIVALUE

Remarks
Before using this property the COD_INIVALUES communication channel must be opened with the OpenData Property. The IniValue property is supported by bar charts only.

Data Type
Float (Single)

See Also
OpenData, CloseData, Passing Data to Chart FX, Value, XValue

Data

Const Porperty

Data

A float property (Single Type in Visual Basic) that sets or returns the value of a constant in the chart, the number of the constant to set (get) is represented by the index supplied.

Visual Basic
```
[form.] Chart1.Const(Index) [ = setting! ]
```

Visual C++
```
fConst = pChart1->GetFloatProperty("Const",Index);
pChart1->SetFloatProperty("Const",fSetting,Index);
```

SQLWindows
```
Set fConst = chart_GetArrNumProp(cc1,"Const",Index)
Call chart_SetArrNumProp(cc1,"Const",fSetting,Index)
```

Borland C++
```
pChart1->GetPropConst(fConst,Index);
pChart1->SetPropConst(fSetting,Index);
```

Property Code
CP_CONST

Remarks
Before using this property the COD_CONSTANTS communication channel must be opened with the OpenData Property.

Data Type
Float (Single)

See Also
OpenData, CloseData, What is the use for constant lines and color stripes?

Colors

ThisColor Property

Data

This property is available at design time so you can pre-set foreground colors to the chart. This Color property will pop-up the colors palette so you can set colors to the Series and/or Points pointed by ThisSerie and ThisPoint property.
A very used method is to set the AutoInc property to TRUE and assign colors to the different series in the Chart. The AutoInc property will automatically increase the series index after setting the color for such series.

See Also
AutoInc, ThisSerie, ThisPoint, ThisBkColor

Data

Data

Color Property

A color (Long) value that sets or returns the color that will be used to paint the markers corresponding to the series supplied as the index of the property.

Visual Basic

```
[form.] Chart1.Color(Index) [ = setting& ]
```

Visual C++

```
lColor = pChart1->GetNumProperty("Color",Index);
pChart1->SetNumProperty("Color",lSetting,Index);
```

SQLWindows

```
Set lColor = chart_GetArrNumProp(cc1,"Color",Index)
Call chart_SetArrColorProp(cc1,CP_COLOR,lSetting,Index)
```

Borland C++

```
pChart1->GetPropColor(lColor,Index);
pChart1->SetPropColor(lSetting,Index);
```

Property Code
CP_COLOR

Remarks
Before using this property the COD_COLORS communication channel must be opened with the OpenData Property.

Data Type
Color (Long)

See Also
OpenData, CloseData, changing default series colors used by Chart FX

Data

ThisBkColor Property

Data

This property is available at design time so you can pre-set background colors to the chart. ThisBKColor property will pop-up the colors palette so you can set background colors to the Series and/or Points pointed by ThisSerie and ThisPoint property.
A very used method is to set the AutoInc property to TRUE and assign colors to the different series in the Chart. The AutoInc property will automatically increase the series index after setting the color for such series.

See Also
AutoInc, ThisSerie, ThisPoint, ThisColor

Data

Data

BkColor Property

A color (Long) value that sets or returns the background color that will be used to paint the markers corresponding to the series supplied as the index of the property.

Visual Basic

```
[form.] Chart1.BkColor(Index) [ = setting& ]
```

Visual C++

```
lBkColor =
pChart1->GetNumProperty("BkColor",Index);
pChart1->SetNumProperty("BkColor",lSetting,Index);
```

SQLWindows

```
Set lBkColor = chart_GetArrNumProp(cc1,"BkColor",Index)
Call chart_SetArrColorProp(cc1,CP_BKCOLOR,lSetting,Index)
```

Borland C++

```
pChart1->GetPropBkColor(lBkColor,Index);
pChart1->SetPropBkColor(lSetting,Index);
```

Property Code
CP_BKCOLOR

Remarks
Before using this property the COD_COLORS communication channel must be opened with the OpenData Property.

Data Type
Color (Long)

See Also
OpenData, CloseData, Passing Data to Chart FX

AdmDlg Property

This property is available at design time so you can set all values related to the Adm property.

See Also
Adm property

Adm Property

A float property (Single Type in Visual Basic) that gets administration values of the chart, the index supplied specify the related value.

Visual Basic
```
[form.] Chart1.Adm(Index) = setting!
```

Visual C++
```
pChart1->SetFloatProperty("Adm",fSetting,Index);
```

SQLWindows
```
Call chart_SetArrNumProp(cc1,"Adm",fSetting,Index)
```

Borland C++
```
pChart1->SetPropAdm(fSetting,Index);
```

Property Code
CP_ADM

Remarks
The indexes that can be used with this property and the meaning of these settings are:

CSA_MIN	Change the minimum value used on the Y-axis
CSA_MIN2	Same as CSA_MIN for the secondary Y axis
CSA_MAX	Change the maximum value used on the Y-axis
CSA_MAX2	Same as CSA_MAX for the secondary Y axis.
CSA_GAP	Change the gap used on the Y-axis.
CSA_GAP2	Same as CSA_GAP for the secondary Y Axis.
CSA_SCALE	Change scale that is used. This constant is very useful when the values used on the Y-axis are too big (i.e. 10.000.000,oo) in this case you can use a 1.000.000 scale and the values on the Y-axis will be divided by this scale.
CSA_SCALE2	Same as CSA_SCALE for the secondary Y axis.
CSA_XSCALE	Same as CSA_SCALE for the secondary X axis (Scatter only)
CSA_YLEYGAP	This constant is used with theYLeg Property in order to change the equivalent units of a y legend text. Please refer to YLeg Property.
CSA_PIXXVALUE	This constant is used with the PixFactor Property to change the equivalent unit representation of Y-axis in pixels. Please refer to PixFactor Property.
CSA_XMIN	Change the minimum value used on the X-axis (scatter)
CSA_XMAX	Change the maximum value used on the X-axis (scatter)
CSA_XGAP	Change the gap used on the X-axis for Scatter Charts. This value is calculated automatically each time you create a chart.
CSA_LOGBASE	Change the Log base used in the Primary Y axis
CSA_LOGBASE2	Same as CSA_LOGBASE for the secondary Y axis
CSA_LOGBASEX	Same as CSA_LOGBASE for the X axis (Scatter only)

Data Type
Float (Single)

Visual

LeftGap Property

Data

An integer value (16 bits) that sets or returns the gap between the specific border of the chart and the.The default value is 40.

Visual Basic
```
[form.] Chart1.LeftGap [ = setting% ]
```

Visual C++
```
nGap = pChart1->GetNumProperty("LeftGap");
pChart1->SetNumProperty("LeftGap",nSetting);
```

SQLWindows
```
Set nGap = chart_GetNumProp(cc1,"LeftGap")
Call chart_SetNumProp(cc1,"LeftGap",nSetting)
```

Borland C++
```
pChart1->GetPropLeftGap(nGap);
pChart1->SetPropLeftGap(nSetting);
```

Remarks

This value is measured in device units (Pixels). The end user can modify this distance manually from the chart window by pointing the mouse near each border and dragging it to the desired position. to stop the user from changing this distance please refer to Style Property with CS_RESIZEABLE code.

Data Type

Integer

See Also

TopGap, LeftGap, BottomGap

RightGap Property

An integer value (16 bits) that sets or returns the gap between the specific border of the chart and the.The default value is 40.

Visual Basic

```
[form.] Chart1.RightGap [ = setting% ]
```

Visual C++

```
nGap = pChart1->GetNumProperty("RightGap");
pChart1->SetNumProperty("RightGap",nSetting);
```

SQLWindows

```
Set nGap = chart_GetNumProp(cc1,"RightGap")
Call chart_SetNumProp(cc1,"RightGap",nSetting)
```

Borland C++

```
pChart1->GetPropRightGap(nGap);
pChart1->SetPropRightGap(nSetting);
```

Remarks

This value is measured in device units (Pixels). The end user can modify this distance manually from the chart window by pointing the mouse near each border and dragging it to the desired position. to stop the user from changing this distance please refer to Style Property with CS_RESIZEABLE code.

Data Type

Integer

See Also

LeftGap, TopGap, BottomGap

TopGap Property

An integer value (16 bits) that sets or returns the gap between the specific border of the chart and the.The default value is 40.

Visual Basic

```
[form.] Chart1.TopGap [ = setting% ]
```

Visual C++

```
nGap = pChart1->GetNumProperty("TopGap");
pChart1->SetNumProperty("TopGap",nSetting);
```

SQLWindows

```
Set nGap = chart_GetNumProp(cc1,"TopGap")
Call chart_SetNumProp(cc1,"TopGap",nSetting)
```

Borland C++

```
pChart1->GetPropTopGap(nGap);
pChart1->SetPropTopGap(nSetting);
```

Remarks

This value is measured in device units (Pixels). The end user can modify this distance manually from the chart window by pointing the mouse near each border and dragging it to the desired position. to stop the user from changing this distance please refer to Style Property with CS_RESIZEABLE code.

Data Type

Integer

See Also

LeftGap, RightGap, BottomGap

BottomGap Property

An integer value (16 bits) that sets or returns the gap between the specific border of the chart and the.The default value is 40.

Visual Basic

```
[form.] Chart1.BottomGap [ = setting% ]
```

Visual C++

```
nGap = pChart1->GetNumProperty("BottomGap");
pChart1->SetNumProperty("BottomGap",nSetting);
```

SQLWindows

```
Set nGap = chart_GetNumProp(cc1,"BottomGap")
Call chart_SetNumProp(cc1,"BottomGap",nSetting)
```

Borland C++

```
pChart1->GetPropBottomGap(nGap);
pChart1->SetPropBottomGap(nSetting);
```

Remarks

This value is measured in device units (Pixels). The end user can modify this distance manually from the chart window by pointing the mouse near each border and dragging it to the desired position. to stop the user from changing this distance please refer to Style Property with CS_RESIZEABLE code.

Data Type

Integer

See Also

TopGap, LeftGap, RightGap

Decimals Property

This property allows you to set the number of decimals in the chart. This property will apply to all chart tools, including data points, axis' and other chart items that have associated values.
If you want to set number of decimals for an specific chart item please refer to DecimalsNum property.

See Also
DecimalsNum

Markers

PointType Property

Data

An integer value (16 bits) that sets or returns the type of point used to paint markers in line, mark, spline and scatter charts. The default value is CHART_RECTMK.

Visual Basic

```
[form.] Chart1.PointType [ = setting% ]
```

Visual C++

```
nPointType = pChart1->GetNumProperty("PointType");
pChart1->SetNumProperty("PointType",nSetting);
```

SQLWindows

```
Set nPointType = chart_GetNumProp(cc1,"PointType")
Call chart_SetNumProp(cc1,"PointType",nSetting)
```

Borland C++

```
pChart1->GetPropPointType(nPointType);
pChart1->SetPropPointType(nSetting);
```

Property Code
CP_POINTTYPE

Remarks
Setting = Pre-defined point style
To display Wingdings or any font as data marker please refer to MultiPoint property.

Data Type
Integer

See Also
MultiPoint, MultiType, MultiShape, Shape

Scheme Property

An integer value (16 bits) that sets or returns the scheme used to paint the markers. The default value is CHART_CSSOLID.

Visual Basic

```
[form.] Chart1.Scheme [ = setting% ]
```

Visual C++

```
nScheme = pChart1->GetNumProperty("Scheme");
pChart1->SetNumProperty("Scheme",nSetting);
```

SQLWindows

```
Set nScheme = chart_GetNumProp(cc1,"Scheme")
Call chart_SetNumProp(cc1,"Scheme",nSetting)
```

Borland C++

```
pChart1->GetPropScheme(nScheme);
pChart1->SetPropScheme(nSetting);
```

Property Code
CP_SCHEME

Remarks
This function is provided to end user through the options dialog.
See Color Schemes for supported values.

Data Type
Integer

See Also
PatternBar, Pattern

Stacked Property

An integer value (16 bits) that sets or returns the type of stack used to draw area and bar charts. The default value is CHART_NOSTACKED.

Visual Basic

```
[form.] Chart1.Stacked [ = setting% ]
```

Visual C++

```
nStacked = pChart1->GetNumProperty("Stacked");
pChart1->SetNumProperty("Stacked",nSetting);
```

SQLWindows

```
Set nStacked = chart_GetNumProp(cc1,"Stacked")
Call chart_SetNumProp(cc1,"Stacked",nSetting)
```

Borland C++

```
pChart1->GetPropStacked(nStacked);
pChart1->SetPropStacked(nSetting);
```

Property Code

CP_STACKED

Remarks

This property affects Bar and Area charts only. This function is provided to the end user through the options dialog.

Comments

CHART_NOSTACKED	Remove stacked option.
CHART_STACKED	Set stacked option
CHART_STACKED100	Set stacked 100% option

Data Type

Integer

Axis

Data

Data

Grid Property

An integer value (16 bits) that sets or returns the type of grid. The default value is CHART_NOGRID.

```
Visual Basic
[form.] Chart1.Grid [ = setting% ]
```

```
Visual C++
nGrid = pChart1->GetNumProperty("Grid");
pChart1->SetNumProperty("Grid",nSetting);
```

```
SQLWindows
Set nGrid = chart_GetNumProp(cc1,"Grid")
Call chart_SetNumProp(cc1,"Grid",nSetting)
```

```
Borland C++
pChart1->GetPropGrid(nGrid);
pChart1->SetPropGrid(nSetting);
```

Property Code
CP_GRID

Comments

CHART_NOGRID	Remove both grids
CHART_HORZGRID	Grid for the Primary Y axis
CHART_VERTGRID	Set vertical grid (X axis)
CHART_GRIDY2	Grid for the Second Y axis
CHART_BOTHGRID	Set both grids (Primary and X)

Remarks
This property does not affect Pie Charts. This function is provided to end user through the menu. See Grid Styles for supported values.

Data Type
Integer

See Also
VertGridGap

Data

WallWidth Property

Data

An integer value (16 bits) that sets or returns the wall's width of a 3D Chart. The default value is 8.

Visual Basic

```
[form.] Chart1.WallWidth [ = setting% ]
```

Visual C++

```
nWidth = pChart1->GetNumProperty("WallWidth");
pChart1->SetNumProperty("WallWidth",nSetting);
```

SQLWindows

```
Set nWidth = chart_GetNumProp(cc1,"WallWidth")
Call chart_SetNumProp(cc1,"WallWidth",nSetting)
```

Borland C++

```
pChart1->GetPropWallWidth(nWidth);
pChart1->SetPropWallWidth(nSetting);
```

Property Code
CP_WALLWIDTH

Remarks
This property does not affect 2D Charts and is measured in device units (Pixels).

Data Type
Integer

Pattern Property

An integer value (16 bits) that sets the pattern used to paint the serie supplied as an index of the property.
This value must be less than 16 and represents the index of the
pattern in the current PatternBar.

Visual Basic

```
[form.] Chart1.Pattern(Index) = setting%
```

Visual C++

```
pChart1->SetNumProperty("Pattern",nSetting,index);
```

SQLWindows

```
Call chart_SetArrNumProp(cc1,"Pattern",nSetting,Index)
```

Borland C++

```
pChart1->SetPropPattern(nSetting,Index);
```

Property Code
CP_PATTERN

Remarks
This property affects charts with pattern schemes only (See Scheme Property)

Data Type
Integer

See Also
Scheme, PatternBar

Legends

BarHorzGap Property

Data

An integer value (16 bits) that sets or returns the width or gap of the legend (Text) for a horizontal bar chart. The default value is 50.

Visual Basic

```
[form.] Chart1.BarHorzGap [ = setting% ]
```

Visual C++

```
nGap = pChart1->GetNumProperty("BarHorzGap");
pChart1->SetNumProperty("BarHorzGap",nSetting);
```

SQLWindows

```
Set nGap = chart_GetNumProp(cc1,"BarHorzGap")
Call chart_SetNumProp(cc1,"BarHorzGap",nSetting)
```

Borland C++

```
pChart1->GetPropBarHorzGap(nGap);
pChart1->SetPropBarHorzGap(nSetting);
```

Property Code
CP_BARHORZGAP

Remarks
This property affects only horizontal bar charts and is measured in device units (Pixels).

Data Type
Integer

See Also
RGBBarHorz

Data

ViewRot3d Property

Data

This property is available at design time so you can add special rotation effects to the chart control. This capability is also available from Chart FX Toolbar or tabbed dialog.

See Also
Chart3D, View3D, Angles3D

View3D Property

A Boolean (Integer) value that sets or returns View mode of the chart in 3D. The default value is FALSE.

Visual Basic

```
[form.] Chart1.View3D [ = setting% ]
```

Visual C++

```
bView3D = pChart1->GetNumProperty("View3D");
pChart1->SetNumProperty("View3D",bSetting);
```

SQLWindows

```
Set bView3D = chart_GetNumProp(cc1,"View3D")
Call chart_SetNumProp(cc1,"View3D",bSetting)
```

Borland C++

```
pChart1->GetPropView3D(bView3D);
pChart1->SetPropView3D(bSetting);
```

Property Code
CP_VIEW3D

Remarks
Removing the 3D View (FALSE) sets the axes at right angles independent of chart rotation. Setting the 3D View (TRUE) show axes in perspective.This property is provided to the end user through the rotation dialog.
At design time the programmer must use the ViewRot3D Property

Data Type
Integer

See Also
Chart3D, Angles3D

Angles3D Property

A Long value that sets or returns the angles used to draw the chart in 3DView mode. The X-angle is in the low word and the Y-angle is in the high word. The default value is
0,0.

```
Visual Basic
[form.] Chart1.Angles3D [ = setting& ]
```

```
Visual C++
lAngles = pChart1->GetNumProperty("Angles3D");
pChart1->SetNumProperty("Angles3D",lSetting);
```

```
SQLWindows
Set lAngles = chart_GetNumProp(cc1,"Angles3D")
Call chart_SetNumProp(cc1,"Angles3D",lSetting)
```

```
Borland C++
pChart1->GetPropAngles3D(lAngles);
pChart1->SetPropAngles3D(lSetting);
```

Property Code
CP_ANGLES3D

Remarks
This property is only used when drawing the chart in 3DView mode (when View3D Property is set to TRUE).
This function is provided to the end user through the rotation dialog.
At design time the programmer can use the ViewRot3D Property to set the initial values for these angles.

Data Type
Long

See Also
Chart3D, View3D, ViewRot3D

Data

PixFactor Property

Data

An Integer value (16 bits) that sets or returns the equivalent unit representation of Y-axis in pixels. This means, that you can change the factor in pixels for each unit in the
Y-axis, to accomplish vertical scroll bars in the chart window. The default value is 0.

Visual Basic
```
[form.] Chart1.PixFactor [ = setting% ]
```

Visual C++
```
nPix = pChart1->GetNumProperty("PixFactor");
pChart1->SetNumProperty("PixFactor",nSetting);
```

SQLWindows
```
Set nPix = chart_GetNumProp(cc1,"PixFactor")
Call chart_SetNumProp(cc1,"Pixfactor",nSetting)
```

Borland C++
```
pChart1->GetPropPixFactor(nPix);
pChart1->SetPropPixFactor(nSetting);
```

Property Code
CP_PIXFACTOR

Remarks
This property is used in conjunction with the Adm Property (CSA_PIXXVALUE code) to modify the pixel factor of the unit in Y-axis. A value of zero (0) means the library will choose an appropiate PixFactor for the chart to fit the current height.

Data Type
Integer

See Also
Adm Property

LineWidth Property

Obsolete API from Chart FX 2.0

This property has been replaced for ItemWidth using the CI_2DLINE constant. This property used to change the line width for a 2D Line Chart.

Important Note:
The MultiLineStyle property has been introduced to support different line styles in a 2D Line chart

LineStyle Property

Obsolete API from Chart FX 2.0

This property has been replaced for ItemStyle using the CI_2DLINE constant. This property used to change the line style for a 2D Line Chart.

Important Note:
The MultiLineStyle property has been introduced to support different line styles in a 2D Line chart

LineColor Property

Obsolete API from Chart FX 2.0

This property has been replaced for ItemColor using the CI_2DLINE constant. This property used to change the line color for a 2D Line Chart.

Important Note:
The MultiLineStyle property has been introduced to support different line styles in a 2D Line chart

LineBKColor Property

Obsolete API from Chart FX 2.0

This property has been replaced for ItemBkColor using the CI_2DLINE constant. This property used to change the back color of the lines for a 2D Line Chart.

Other

FixedWidth Property

Data

Obsolete API from Chart FX 2.0

Data This property has been replaced for ItemWidth using the CI_FIXED constant. This property used to change the width of the constant lines displayed in the chart.

FixedStyle Property

Obsolete API from Chart FX 2.0

This property has been replaced for ItemStyleusing the CI_FIXED constant. This property used to change the style of the constant lines displayed in the chart.

FixedColor Property

Obsolete API from Chart FX 2.0

This property has been replaced for ItemColor using the CI_FIXED constant. This property used to change the color of the constant lines displayed in the chart.

FixedBKColor Property

Obsolete API from Chart FX 2.0

This property has been replaced for ItemBkColor using the CI_FIXED constant. This property used to change the back color of the constant lines displayed in the chart.

FixedGap Property

An Integer value (16 bits) that sets or returns the minimum separation of each unit in the X-axis (in pixels) where 0 means a default value choosen by the library. The default value is 0.

Visual Basic

```
[form.] Chart1.FixedGap [ = setting% ]
```

Visual C++

```
nGap = pChart1->GetNumProperty("FixedGap");
pChart1->SetNumProperty("FixedGap",nSetting);
```

SQLWindows

```
Set nGap = chart_GetNumProp(cc1,"FixedGap")
Call chart_SetNumProp(cc1,"FixedGap",nSetting)
```

Borland C++

```
pChart1->GetPropFixedGap(nGap);
pChart1->SetPropFixedGap(nSetting);
```

Property Code
CP_FIXEDGAP

Remarks
This property is measured in device units (Pixels).

Data Type
Integer

DblClk Dlg Property

This property is available at design time so you can specify how Chart FX will respond to double-click events triggered by the end users. Default behavior is to show a ballon help when the user double-clicks a marker in the chart.
You can set this property to show a balloon help, a dialog or ignore the double-click event.

See Also
Changing Default Balloon Help Text, DblClk Property

DblClk Property

An Integer value (16 bits) that sets the response of the library when the user makes a double click on a marker. The index will specify the type of response and the value must always be zero (0) except in the case of CHART_MENUCLK index where the value is the handle of the popup menu. The default value is CHART_BALLOONCLK.

Visual Basic
```
[form.] Chart1.DblClk(Index) = setting%
```

Visual C++
```
pChart1->SetNumProperty("DblClk",nSetting,Index)
```

SQLWindows
```
Call chart_SetArrNumProp(cc1,"DblClk",nSetting,Index)
```

Borland C++
```
pChart1->SetPropDblClk(nSetting,Index);
```

Property Code
CP_DBLCLK

Comments
CHART_BALLOONCLK = Display a balloon help
CHART_DIALOGCLK = Display a predefined Dialog
CHART_NONECLK = Nothing
CHART_MENUCLK = Display a menu

The library will always notify you with the CN_LBUTTONDBLCLK message when you specify any of the above constants. You may process this message to handle an event.

If you pass a menu to the library you must remember to destroy it when you are finished with it.

Data Type
Integer

See Also
DblClk Dlg, Handling Notification Messages

Data

RigClk Dlg Property

Data

This property is available at design time so you can specify how Chart FX will respond to right click events triggered by the end users. Default behavior is to ignore right click events.
You can set this property to show a balloon help, a dialog or ignore the right-click event.

See Also
Customizing the right click event to show a selection menu, RigClk Property

RigClk Property

An Integer value (16 bits) that sets the response of the library when the user makes a right click on a marker. The index will specify the type of response and the value
must always be zero (0) except in the case of CHART_MENUCLK index where the value is the handle of the popup menu. The default value is CHART_BALLOONCLK.

Visual Basic
```
[form.] Chart1.RigClk(Index) = setting%
```

Visual C++
```
pChart1->SetNumProperty("RigClk",nSetting,Index)
```

SQLWindows
```
Call chart_SetArrNumProp(cc1,"RigClk",nSetting,Index)
```

Borland C/C++
```
pChart1->SetPropRightClk(nSetting,Index);
```

Property Code
CP_RIGCLK

Comments

CHART_BALLOONCLK	Display a balloon help
CHART_DIALOGCLK	Display a predefined dialog
CHART_NONECLK	Nothing
CHART_MENUCLK	Display a menu

Remarks
See Click Styles for index supported values. Note that the library will always generate the RigClk event. If you pass a menu to the library you must remember to destroy it when you are finished with it.

Data Type
Integer

See Also
RigClk Dlg, Handling Notification Messages

RGBBarHorz Property

A Color (Long) value that sets or returns the color of the X legend background of a horizontal Bar Chart. The default value is cyan (RGB(0,255,255))

Visual Basic

```
[form.] Chart1.RGBBarHorz [ = setting& ]
```

Visual C++

```
lColor = pChart1->GetNumProperty("RGBBarHorz");
pChart1->SetNumProperty("RGBBarHorz",lSetting);
```

SQLWindows

```
Set lColor = chart_GetNumProp(cc1,"RGBBarHorz")
Call chart_SetColorProp(cc1,"RGBBarHorz",lSetting)
```

Borland C++

```
pChart1->GetPropRGBBarHorz(lColor);
pChart1->SetPropRGBBarHorz(lSetting);
```

Property Code
CP_RGBBARHORZ

Remarks
This function is provided to the end user by allowing him to drag & drop a color from the PaletteBar to any part of the area of this Legend.

Data Type
Long

See Also
BarHorzGap

RGBBk Property

A Color (Long) value that sets or returns the color for the background sorrounding the chart. The default value is light gray (RGB(192,192,192))

Visual Basic

```
[form.] Chart1.RGBBk [ = setting& ]
```

Visual C++

```
lColor = pChart1->GetNumProperty("RGBBk");
pChart1->SetNumProperty("RGBBk",lSetting)
```

SQLWindows

```
Set lColor = chart_GetNumProp(cc1,"RGBBk")
Call chart_SetColorProp(cc1,"RGBBk",lSetting)
```

Borland C++

```
pChart1->GetPropRGBBk(lColor);
pChart1->SetPropRGBBk(lSetting);
```

Property Code
CP_RGBBK

Remarks
This function is provided to the end user by allowing him to drag & drop a color from the PaletteBar to any part of this background.
Important Note: CHART_TRANSPARENT constant can be set to make a transparent background.

Data Type
Long

See Also
RGB2DBk, RGB3DBk

RGB2DBk Property

A Color (Long) value that sets or returns the color for the 2D charts background. The default value is light gray (RGB(192,192,192)).

Visual Basic

```
[form.] Chart1.RGB2DBk [ = setting& ]
```

Visual C++

```
lColor = pChart1->GetNumProperty("RGB2DBk");
pChart1->SetNumProperty("RGB2DBk",lSetting);
```

SQLWindows

```
Set lColor = chart_GetNumProp(cc1,"RGB2DBk")
Call chart_SetColorProp(cc1,"RGB2DBk",lSetting)
```

Borland C++

```
pChart1->GetPropRGB2DBk(lColor);
pChart1->SetPropRGB2DBk(lSetting);
```

Property Code
CP_RGB2DBK

Remarks
This function is provided to the end user by allowing him to drag & drop a color from the PaletteBar to any part of this background.
Important Note: CHART_TRANSPARENT constant can be set to make a transparent 2D background.

Data Type
Long

See Also
RGBBk, RGB3DBk

RGB3DBk Property

A Color (Long) value that sets or returns the color for 3D charts background. The default value is white (RGB(255,255,255)).

Visual Basic

```
[form.] Chart1.RGB3DBk [ = setting& ]
```

Visual C++

```
lColor = pChart1->GetNumProperty("RGB3DBk");
pChart1->SetNumProperty("RGB3DBk",lSetting);
```

SQLWindows

```
Set lColor = chart_GetNumProp(cc1,"RGB3DBk")
Call chart_SetColorProp(cc1,"RGB3DBk",lSetting)
```

Borland C++

```
pChart1->GetPropRGB3DBk(lColor);
pChart1->SetPropRGB3DBk(lSetting);
```

Property Code
CP_RGB3DBK

Remarks
This function is provided to the end user by allowing him to drag & drop a color from the PaletteBar to any part of this background. This property is used by 3D charts only.
Important Note: CHART_TRANSPARENT constant can be set to make a transparent 3D background.

Data Type
Long

See Also
RGBBk, RGB2DBk

Data

Font Dlg Property

Data

This property is available at design time so you can set the font type to any title or object in the Chart.

See Also:
Font, RGBFont

Font Property

A Long value (32 bits) that sets the font used to draw different texts in a chart. The index of this property represents the text to change and the value represents the new
font. The value must always be a combination (bitwise OR) of Font Styles.

Visual Basic
```
[form.] Chart1.Font(Index) = setting&
```

Visual C++
```
pChart1->SetNumProperty("Font",lSetting,index);
```

SQLWindows
```
Call chart_SetArrNumProp(cc1,"Font",lSetting,Index)
```

Borland C++
```
pChart1->SetPropFont(lSetting,Index);
```

Property Code
CP_FONT

Comments

CHART_LEFTFT	Left Title
CHART_RIGHTFT	Right Title
CHART_TOPFT	Top Title
CHART_BOTTOMFT	Bottom Title
CHART_XLEGFT	X Legend
CHART_YLEGFT	Y Legend
CHART_FIXEDFT	Constants
CHART_LEGENDFT	Legend
CHART_VALUESFT	Values (ShowValues)
CHART_POINTFT	Point Types

Remarks
This function is provided to the end user through the menu.See Font Styles for property supported values.You must take care of giving a TRUE-TYPE font when setting the left or right title.

Data Type
Long

See Also
RGBFont, Font Dlg, HFont, Fonts table

Data

Title Dlg Property

Data

This property is available at design time so you can pre-set any title to the chart. Supported Titles are: Top, Left, Bottom and Right. These titles also support multiline capabilities.

See Also
Font, Title Property

Other

Title Property

Data

A string value that sets or returns the titles of the chart. The type of title to set is specified through the index supplied.

Visual Basic
```
[form.] Chart1.Title(Index) [ = setting$ ]
```

Visual C++
```
pChart1->SetStrProperty("Title",sSetting,index);
```

SQLWindows
```
Call chart_SetArrStrProp(cc1,"Title",nSetting,Index)
```

Borland C++
```
pChart1->SetPropTitle(sSetting,Index);
```

Property Code
CP_TITLE

Remarks
This function is provided to the end user through the menu.

Comments

CHART_LEFTTIT	= Left Title
CHART_RIGHTTIT	= Right Title
CHART_TOPTIT	= Top Title
CHART_BOTTOMTIT	= Bottom Title

Data Type
String

See Also
Title Dlg

Tools

Status Property

Data

A boolean (Integer) value (16 bits) that determines whether the StatusBar is visible or hidden.

Visual Basic

```
[form.] Chart1.ShowStatus = setting%
```

Visual C++

```
pChart1->SetNumProperty("ShowStatus",bSetting);
```

SQLWindows

```
Call chart_SetNumProp(cc1,"ShowStatus",bSetting)
```

Borland C++

```
pChart1->SetPropShowStatus(bSetting);
```

Property Code

CP_SHOWSTATUS

Remarks

This function is provided to the end user through the menu if the statusBar has been previously created by the programmer. See also chapter 5 of the programmer's guide.

Data Type

Integer

See Also

chart_SetStatusItem, How do I create a status bar?

ShowStatus Property

A boolean (Integer) value (16 bits) that determines whether the StatusBar is visible or hidden.

```
Visual Basic
[form.] Chart1.ShowStatus = setting%
```

```
Visual C++
pChart1->SetNumProperty("ShowStatus",bSetting);
```

```
SQLWindows
Call chart_SetNumProp(cc1,"ShowStatus",bSetting)
```

```
Borland C++
pChart1->SetPropShowStatus(bSetting);
```

Property Code
CP_SHOWSTATUS

Remarks
This function is provided to the end user through the menu if the statusBar has been previously created by the programmer. See also chapter 5 of the programmer's guide.

Data Type
Integer

See Also
chart_SetStatusItem, How do I create a status bar?

Data

Language Property

Data

This string property is used to change the current language used by the library. This language is represented by a resource DLL that holds the dialogs,strings and menus needed by Chart FX.

Visual Basic
```
[form.] Chart1.Language = setting$
```

Visual C++
```
pChart1->SetStrProperty("Language",sSetting);
```

SQLWindows
```
Call chart_SetStrProp(cc1,"Language",sSetting)
```

Borland C++
```
pChart1->SetPropLanguage(sSetting);
```

Remarks

The value of the property (String) must be the name of the resource DLL (including path). To support other languages, the package provides the necessary files (RC, DEF, DLG, C) to translate the information and build it to obtain a DLL (Dynamic Link Library) which is the kind of file that this message handles.

For more information on this topic please refer to International Support on previous sections of this manual Loading this DLL follows the Windows standard search for this type of file, therefore you should place the new DLL containing the language support, in your PATH, WINDOWS or SYSTEM directory.

See Also

International Support

Data

HText Property

Data

A string value that sets or returns the text that Chart FX will use in the dialog or balloon generated by the last double click or right click of the mouse. For example, when the user double clicks with the left mouse button a point in any of the series of the chart a default text containing the Series Name-Legend-point value is automatically
generated, if you want to modify this default text, use this property in conjunction with the DblClk or RigClk events.

Visual Basic
```
[form.] Chart1.HText [ = setting$ ]
```

Visual C++
```
s = pChart1->GetStrProperty("HText");
pChart1->SetStrProperty("HText",sSetting);
```

SQLWindows
```
Call chart_GetStrProp(cc1,sText,"HText")
Call chart_SetStrProp(cc1,"HText",sSetting)
```

Borland C++
```
pChart1->GetPropHText(sText);
pChart1->SetPropHText(sSetting);
```

Property Code
CP_HTEXT

Data Type
String

See Also
Handling Notification messages, Changing the default text in the balloon

Data

Legend Property

Data

A string value that sets or returns the text of the X Legend, the index supplied specifies the position of the legend.

Visual Basic

```
[form.] Chart1.Legend(Index) [ = setting$ ]
```

Visual C++

```
sText = pChart1->GetStrProperty("Legend",Index);
pChart1->SetStrProperty("Language",sSetting,Index);
```

SQLWindows

```
Call chart_GetArrStrProp(cc1,sText,"Legend",Index)
Call chart_SetArrStrProp(cc1,"Legend",sSetting,Index)
```

Borland C++

```
pChart1->GetPropLegend(sText,Index);
pChart1->SetPropLegend(sSetting,Index);
```

Property Code
CP_LEGEND

Remarks
Normally you will supply as many legends as the number of points contained in the chart.

Data Type
String

See Also
SerLeg, KeyLeg, KeySer, Assign X and series legends to the chart

Data

SerLeg Property

Data

A string value that sets or returns the text of the Series, the index supplied specifies the position of the legend.

Visual Basic
```
[form.] Chart1.SerLeg(Index) [ = setting$ ]
```

Visual C++
```
sText = pChart1->GetStrProperty("SerLeg",Index);
pChart1->SetStrProperty("SerLeg",sSetting,Index);
```

SQLWindows
```
Call chart_GetArrStrProp(cc1,sText,"SerLeg",Index)
Call chart_SetArrStrProp(cc1,"SerLeg",sSetting,Index)
```

Borland C++
```
pChart1->GetPropSerLeg(sText,Index);
pChart1->SetPropSerLeg(sSetting,Index);
```

Property Code
CP_SERLEG

Remarks
Normally you will supply as many legends as the number of series of the chart.

Data Type
String

See Also
Legend, KeySer, KeyLeg, Assign X and series legends to the chart

Data

KeyLeg Property

Data

A string value that sets or returns the text of the X Legend Keys, the index supplied specifies the position of the legend.

Visual Basic

```
[form.] Chart1.KeyLeg(Index) [ = setting$ ]
```

Visual C++

```
sText = pChart1->GetStrProperty("KeyLeg",Index);
pChart1->SetStrProperty("KeyLeg",sSetting,Index);
```

SQLWindows

```
Call chart_GetArrStrProp(cc1,sText,"KeyLeg",Index)
Call chart_SetArrStrProp(cc1,"KeyLeg",sSetting,Index)
```

Borland C++

```
pChart1->GetPropKeyLeg(sText,Index);
pChart1->SetPropKeyLeg(sSetting,Index);
```

Property Code
CP_KEYLEG

Remarks
Normally you will supply as many legends as the number of points of the chart. The library will always draw this Key Legends assuming they are short enough.

Data Type
String

See Also
Legend, SerLeg, KeySer, Assign X and Series legend to the chart

Data

FixLeg Property

Data

A string value that sets or returns the text of the Constant lines, the index supplied specifies the position of the legend.

Visual Basic

```
[form.] Chart1.FixLeg(Index) [ = setting$ ]
```

Visual C++

```
sText = pChart1->GetStrProperty("FixLeg",Index);
pChart1->SetStrProperty("FixLeg",sSetting,Index);
```

SQLWindows

```
Call chart_GetArrStrProp(cc1,sText,"FixLeg",Index)
Call chart_SetArrStrProp(cc1,"FixLeg",sSetting,Index)
```

Borland C++

```
pChart1->GetPropFixLeg(sText,Index);
pChart1->SetPropFixLeg(sSetting,Index);
```

Property Code
CP_FIXLEG

Remarks
Normally you will supply as many legends as the number of constants of the chart. This is the same number that you supplied for the OpenData Property with the COD_CONSTANTS code.
See also Const Property.

Data Type
String

See Also
What is the use for constant lines and color stripes?

Data

Data

Data

YLeg Property

A string value that sets or returns the text of the Y Legend, the index supplied specifies the position of the legend.

Visual Basic
```
[form.] Chart1.YLeg(Index) [ = setting$ ]
```

Visual C++
```
sText = pChart1->GetStrProperty("YLeg",Index);
pChart1->SetStrProperty("YLeg",sSetting,Index);
```

SQLWindows
```
Call chart_GetArrStrProp(cc1,sText,"YLeg",Index)
Call chart_SetArrStrProp(cc1,"YLeg",sSetting,Index)
```

Borland C++
```
pChart1->GetPropYLeg(sText,Index);
pChart1->SetPropYLeg(sSetting,Index);
```

Property Code
CP_YLEG

Remarks
This property mus be used in conjunction with Adm Property with CSA_YLEGGAP code. The value supplied with Adm property is the gap the library will use between the legends.

Data Type
String

See Also
Adm property

Data

KeySer Property

Data

A string value that sets or returns the text of the Series Legend Keys, the index supplied specifies the position of the legend.

Visual Basic
```
[form.] Chart1.KeySer(Index) [ = setting$ ]
```

Visual C++
```
sText = pChart1->GetStrProperty("KeySer",Index);
pChart1->SetStrProperty("KeySer",sSetting,Index);
```

SQLWindows
```
Call chart_GetArrStrProp(cc1,sText,"KeySer",Index)
Call chart_SetArrStrProp(cc1,"KeySer",sSetting,Index)
```

Borland C++
```
pChart1->GetPropKeySer(sText,Index);
pChart1->SetPropKeySer(sSetting,Index);
```

Property Code
CP_KEYSER

Remarks
Normally you will supply as many legends as the number of series of the chart. The library will always draw this Key Legends assuming they are short enough.
This property is supported by horizontal bar charts only.

Data Type
String

See Also
Legend, SerLeg, KeyLeg, Assign X and series legend to the chart

StatusText Property

A string value that sets or returns the text of an existent status item. The index supplied represent the code (ID) of the item.

Visual Basic

```
[form.] Chart1.StatusText(Index) [ = setting$ ]
```

Visual C++

```
sText = pChart1-> GetStrProperty("StatusText",Index);
pChart1-> SetStrProperty("StatusText",sSetting,Index);
```

SQLWindows

```
Call chart_GetArrStrProp(cc1,sText,"StatusText",Index)
Call chart_SetArrStrProp(cc1,"StatusText",sSetting,Index)
```

Borland C++

```
pChart1->GetPropStatusText(sText,Index);
pChart1->SetPropStatusText(sSetting,Index);
```

Property Code
CP_STATUSTEXT

Remarks
This property must be used only after you create the status bar with Status Property or chart_SetStatusItem

Data Type
String

See Also
ShowStatus, How do I create a status bar in the chart?

ExportFile Property

Obsolete API from Chart FX 2.0

This property has been embbeded in the Export property using the CHART_CFXFILE constant. This property used to save (including data) the chart to a file.

ImportFile Property

Obsolete API from Chart FX 2.0

This property has been embbeded in the Import property using the CHART_CFXFILE constant. This property used to save (including data) the chart to a file.

WriteTemplate Property

Obsolete API from Chart FX 2.0

This property has been embbeded in the Export property using the CHART_CFXTEMPLATE constant. This property used to save the chart attributes or template (without data) to a file.

ReadTemplate Property

Obsolete API from Chart FX 2.0

This property has been embbeded in the Import property using the CHART_CFXTEMPLATE constant. This property used to save the chart attributes or template (without data) to a file.

Data

ChartStatus Property

Data

RGBFont Property

A Color (Long) value that sets the color of the font used to draw different text elements in a chart. The index of this property represents the text to change and the value represents the new color.

Visual Basic
```
[form.] Chart1.RGBFont(Index) = setting&
```

Visual C++
```
pChart1->SetNumProperty("RGBFont",lSetting,index);
```

SQLWindows
```
Call chart_SetArrNumProp(cc1,"RGBFont",lSetting,Index)
```

Borland C++
```
pChart1->SetPropRGBFont(lSetting,Index);
```

Property Code
CP_RGBFONT

Remarks
This function is provided to the end user through the menu. At design time the user can change the colors through the FontDlg Property.

Comments
You may change any of the color of the text of titles shown below:

CHART_LEFTFT	Left title
CHART_RIGHTFT	Right title
CHART_TOPFT	Top title
CHART_BOTTOMFT	Bottom title
CHART_XLEGFT	X legend
CHART_YLEGFT	Y legend
CHART_FIXEDFT	Constants
CHART_LEGENDFT	Legend
CHART_VALUESFT	Values (ShowValues)
CHART_POINTFT	Point Types
CHART_Y2LEGFT	Secondary Y Legend

Data Type
Long

See Also
Font, HFont, Title

Edit Property

This is a read-only property that holds a window handle when the user is trying to change a legend or value in the DataEditor.

Visual Basic
```
[form.] Chart1.Edit
```

Visual C++
```
hWnd = pChart1->GetNumProperty("Edit");
```

SQLWindows
```
Set hWnd = chart_GetNumProp(cc1,"Edit")
```

Borland C++
```
pChart1->GetPropEdit(hWnd);
```

Property Code
CP_EDIT

Remarks
This property must only be used in response of a notification of the ChangeString Event or the ChangeValue Event.

Data Type
Integer

See Also
ChangeString event, Handling Notification Messages.

CopyData Property

Obsolete API from Chart FX 2.0

This property has been embbeded in the Export property using the CHART_DATA constant. This property used to copy the Data to the clipboard using the TSV (Tab Separated Values) format.

CopyBitmap Property

Obsolete API from Chart FX 2.0

This property has been embbeded in the Export property using the CHART_BITMAP constant.
This property used to copy the chart Picture to the clipboard using the BITMAP format.

PrintIt Property

This property allows the programmer to print the chart.
The value assigned to this property is not used (must be set to zero).

Visual Basic

```
[form.] Chart1.PrintIt = setting%
```

Visual C++

```
pChart1->SetNumProperty("PrintIt", setting%);
```

SQLWindows

```
Call chart_SetNumProp(cc1,"PrintIt", setting%)
```

Borland C++

```
pChart1->SetPropPrintIt(setting%);
```

Property Code
CP_PRINT

Remarks
Setting:
0 = if printing the whole chart.
LOWORD = From (0 for beggining)
HIWORD = To (0 for last)
This function is provided to the end user through the menu if permitted by the programmer by using the CS_PRINTABLE style.

Data Type
Long

See Also
chart_Paint, How do I print several charts in the same page?

HFont Property

An Integer value (16 bits) that sets the font used to draw different text elements in a chart. The index of this property represents the text to change and the value represents
the new font. The value must always be a valid font handle (HFONT)

Visual Basic
```
[form.] Chart1.hFont(Index) = setting%
```

Visual C++
```
pChart1->SetNumProperty("hFont",nSetting,index);
```

SQLWindows
```
Call chart_SetArrNumProp(cc1,"hFont",nSetting,Index)
```

Borland C++
```
pChart1->SetProphFont(nSetting,Index);
```

Property Code
CP_HFONT

Remarks
This function is provided to the end user through the menu. You must take care of destroying the font passed to this property.

Comments
You may change any of the color of the text of titles shown below

CHART_LEFTFT	Left title
CHART_RIGHTFT	Right title
CHART_TOPFT	Top title
CHART_BOTTOMFT	Bottom title
CHART_XLEGFT	X legend
CHART_YLEGFT	Y legend
CHART_FIXEDFT	Constants
CHART_LEGENDFT	Legend
CHART_VALUESFT	Values (ShowValues)
CHART_POINTFT	Point Types

Data Type
Integer

See Also
Font, RGBFont, Fonts Table

ThisSerie Property

This property allows you to set/get the actual series when passing information to the chart. Before using the Vlue property or other data related property you must first set this property to the series you want to pass data to.

Visual Basic

```
[form.] Chart1.ThisSerie = setting%
```

Visual C++

```
pChart1->SetStrProperty("ThisSerie",sSetting);
```

SQLWindows

```
Call chart_SetStrProp(cc1,"ThisSerie",sSetting)
```

Borland C++

```
pChart1->SetPropThisSerie(sSetting);
```

Property Code
CP_THISSERIE

Remarks
Setting is the series index (Zero Based)

Data Type
Integer

See Also
Passing Data to Chart FX, Value Property

ChartType Property

This property allows you to change the chart type used. This property is used to change the chart type itself, without taking into consideration other CT_ constants. This property is commonl used at design time to select the chart type desired in your form.

Visual Basic
```
[form.] Chart1.ChartType = setting%
```

Visual C++
```
pChart1->SetStrProperty("ChartType",sSetting);
```

SQLWindows
```
Call chart_SetStrProp(cc1,"ChartType",sSetting)
```

Borland C++
```
pChart1->SetPropChartType(sSetting);
```

Property Code
CP_CHARTTYPE

Remarks
Setting can be any of the following:

LINE	Line Chart
BAR	Bar Chart (Including Horizontal, and stacked charts)
SPLINE	Curve-fitting Chart
MARK	Point Chart
PIE	Pie Chart
AREA	Area Chart (Including stacked charts)
PARETO	Pareto Chart (Statistical Chart. Special)
SCATTER	Scatter Chart
HILOW	Hi-Low Close Chart
SURFACE	Surface Charts
POLAR	Polar Charts (also in 3D!)
CUBE	Cube Charts
DOUGHNUT	Doughnut Charts

Important Note: If you want to include other CT_ constant in your chart type, please refer to the Type property that will allow you to change or include CT_ constants in your chart control

Data Type
Integer

See Also
Type, TypeEx, Style, StyleEx

Chart3D Property

This property allows you to turn on/off 3D effect to the chart. Available at design or run time

Visual Basic

```
[form.] Chart1.Chart3D = setting%
```

Visual C++

```
pChart1->SetStrProperty("Chart3D",sSetting);
```

SQLWindows

```
Call chart_SetStrProp(cc1,"Chart3D",sSetting)
```

Borland C++

```
pChart1->SetPropChart3D(sSetting);
```

Property Code
CP_CHART3D

Remarks
Setting
TRUE = Turn on 3D effect
FALSE = Turn off 3D effect

Data Type
Boolean

See Also
View3D, Angles3D, ViewRot3D

Data

ToolBar Property

Data

This property allows you to turn on/off the toolbar in the chart Control.

Visual Basic
```
[form.] Chart1.ToolBar = setting%
```

Visual C++
```
pChart1->SetStrProperty("ToolBar",sSetting);
```

SQLWindows
```
Call chart_SetStrProp(cc1,"ToolBar",sSetting)
```

Borland C++
```
pChart1->SetPropToolbar(sSetting);
```

Property Code
CP_TOOLBAR

Remarks
Setting
TRUE = Show Toolbar
FALSE =Hide Toolbar

Data Type
Boolean

See Also
PaletteBar, PatternBar, Customizing the Toolbar

PaletteBar Property

This property allows you to turn on/off the PaletteBar in the chart Control.

Visual Basic

```
[form.] Chart1.PaletteBar = setting%
```

Visual C++

```
pChart1->SetStrProperty("PaletteBar",sSetting);
```

SQLWindows

```
Call chart_SetStrProp(cc1,"PaletteBar",sSetting)
```

Borland C++

```
pChart1->SetPropPaletteBar(sSetting);
```

Property Code
CP_PALETTEBAR

Remarks
Setting
TRUE = Show Palettebar
FALSE =Hide Palettebar

Data Type
Boolean

See Also
Toolbar , PatternBar

Data

PatternBar Property

Data

This property allows you to turn on/off the PatternBar in the chart Control.

Visual Basic
```
[form.] Chart1.PatternBar = setting%
```

Visual C++
```
pChart1->SetStrProperty("PatternBar",sSetting);
```

SQLWindows
```
Call chart_SetStrProp(cc1,"PatternBar",sSetting)
```

Borland C++
```
pChart1->SetPropPatternbar(sSetting);
```

Property Code
CP_PATTERNBAR

Remarks
Setting
TRUE = Show Patternbar
FALSE =Hide Patternbar

Data Type
Boolean

See Also
PaletteBar, Toolbar

Data

ThisPoint Property

Data

This property is avaiable at design time so you can pre-set colors or any other information to the index pointed by the ThisPoint property.

See Also
ThisSerie, ThisColor, ThisBKColor

CustTool Property

Data

Obsolete API from Chart FX 2.0

Data This property has been replaced by the CustomTool property. This property used to specify which buttons appeared in the Toolbar. Since Chart FX Toolbar now includes Icon Combos (Gallery and PaletteBar) this property is now useless.

HctlWnd Property

Data

This property allows you to retrieve the window handle for the chart control, since some development tools do not provide this handle as a standard property

Visual Basic

```
[form.] Chart1.HCtlWnd
'Supported in Visual Basic as Chart1.hWnd
```

Visual C++

```
pChart1->GetNumProperty("HCtlWnd");
```

SQLWindows

```
Call chart_GetStrProp(cc1,"HCtlWnd")
```

Borland C++

```
pChart1->GetPropHCtlWnd();
```

Property Code
CP_HCTLWND

Remarks
This property is very useful when you want to perform any action to the chart window manually and therefore you need the chart window handle to perform this action

Data Type
Boolean

See Also
chart_Send

ReturnValue Property

Data

Reserved property used for compatibility among different development tools. ***Do not use!***

Reserved1 Property

Data

Reserved property used for compatibility among different development tools. ***Do not use!***

Reserved2 Property

Data

Reserved property used for compatibility among different development tools. ***Do not use!***

Reserved3 Property

Data

Reserved property used for compatibility among different development tools. ***Do not use!***

AutoInc Property

Data

This property is really useful when you're setting colors to the chart at design time. When turned on this property will automatically increase the Point and/or Series so you don't have to return and set those properties to the appropiate Point/Series Index.

ThisValue Property

Data

Data

VertGridGap Property

Data

An Integer value (16 bits) that sets or returns the vertical grid gap. This is the distance (measured in X points) between 2 lines of the vertical grid

Visual Basic

```
[form.] Chart1.VertGridGap [ = setting! ]
```

Visual C++

```
fConst = pChart1->GetNumProperty("VertGridGap");
pChart1->SetNumProperty("VertGridGap",fSetting);
```

SQLWindows

```
Set fConst = chart_GetArrNumProp(cc1,"VertGridGap")
Call chart_SetArrNumProp(cc1,"VertGridGap",fSetting)
```

Borland C++

```
pChart1->GetPropVertGridGap(fConst);
pChart1->SetPropVertGridGap(fSetting);
```

Property Code

CP_VERTGRIDGAP

Remarks

The value must be greater than 0

When setting to a positive value the grid will start at point specify in wParam. On the other hand, if you set wParam to a negative value, a Grid line will always be placed in Point No 1. and the next one separated with the abs value of wParam.

Data Type

Integer

See Also

Grid

XLegType Property

Obsolete API from Chart FX 2.0

This property has been embbeded in the LegStyle property. This property used to set different styles for X axis Labels.

ConstType Property

An Integer value (16 bits) that sets or returns the type of the constant values and legends. This property affects how these elements are presented in a chart.

Visual Basic

```
[form.] Chart1.ConstType(Index) [ = setting! ]
```

Visual C++

```
fConst = pChart1-> GetNumProperty("ConstType",Index);
pChart1-> SetNumProperty("ConstType",fSetting,Index);
```

SQLWindows

```
Set fConst = chart_GetArrNumProp(cc1,"ConstType",Index)
Call chart_SetArrNumProp(cc1,"ConstType",fSetting,Index)
```

Borland C++

```
pChart1->GetPropConstType(fConst,Index);
pChart1->SetPropConstType(fSetting,Index);
```

Property Code
CP_CONSTTYPE

Remarks
Index:
Constant index as specify in the Const property

Setting: must be a combination of the following flags:

CC_HIDETEXT	Hide the constants text.
CC_HIDE	Hide the constants.

Data Type
Integer

See Also
Const, FixLeg, What is the use for constant lines and color stripes?

ItemWidth Property

An integer value that sets or returns the width used to paint the item specified in the index.

Visual Basic
```
[form.] Chart1.ItemWidth(Index) [ = setting& ]
```

Visual C++
```
lColor = pChart1-> GetNumProperty("ItemWidth",Index);
pChart1-> SetNumProperty("ItemWidth",lSetting,Index);
```

SQLWindows
```
Set lColor = chart_GetArrNumProp(cc1,"ItemWidth",Index)
Call chart_SetArrNumProp(cc1,"ItemWidth",lSetting,Index)
```

Borland C++
```
pChart1->GetPropItemWidth(lColor,Index);
pChart1->SetPropItemWidth(lSetting,Index);
```

Property Code
CP_ITEMWIDTH

Comments

CI_HORZGRID	Horizontal grid
CI_VERTGRID	Vertical grid
CI_2DLINE	2D Line Chart
CI_FIXED	Constants
CI_LOOPPOS	Loop marker (RealTime)
CI_HORZGRID2	Grid Second Y Axis

Setting:
Width in device units (pixels)

Data Type
Integer

See Also
ItemColor, ItemStyle, MultiLineStyle

ItemStyle Property

An integer value that sets or returns the style used to paint the item specified in the index.

```
Visual Basic
[form.] Chart1.ItemStyle(Index) [ = setting& ]
```

```
Visual C++
lColor = pChart1-> GetNumProperty("ItemStyle",Index);
pChart1-> SetNumProperty("ItemStyle",lSetting,Index);
```

```
SQLWindows
Set lColor = chart_GetArrNumProp(cc1,"ItemStyle",Index)
Call chart_SetArrNumProp(cc1,"ItemStyle",lSetting,Index)
```

```
Borland C++
pChart1->GetPropItemStyle(lColor,Index);
pChart1->SetPropItemStyle(lSetting,Index);
```

Property Code
CP_ITEMSTYLE

Line Styles

Constant	Style Description
CHART_SOLID	Solid Pen
CHART_DASH	Dashed Pen
CHART_DOT	Dotted Pen
CHART_DASHDOT	Dash-Dotted Pen
CHART_DASHDOTDOT	Dash-Dot-Dotted Pen
CHART_PSTRANSPARENT	Transparent Pen (Must be used with other style)

Data Type
Integer

See Also
ItemColor, ItemWidth, MultiLineStyle

ItemColor Property

A Color (Long) value that sets or returns the foreground color used to paint the item specified in the index.

Visual Basic

```
[form.] Chart1.ItemColor(Index) [ = setting& ]
```

Visual C++

```
lColor = pChart1-> GetNumProperty("ItemColor",Index);
pChart1-> SetNumProperty("ItemColor",lSetting,Index);
```

SQLWindows

```
Set lColor = chart_GetArrNumProp(cc1,"ItemColor",Index)
Call chart_SetArrNumProp(cc1,"ItemColor",lSetting,Index)
```

Borland C++

```
pChart1->GetPropItemColor(lColor,Index);
pChart1->SetPropItemColor(lSetting,Index);
```

Property Code
CP_ITEMCOLOR

Comments

CI_HORZGRID	Horizontal grid
CI_VERTGRID	Vertical grid
CI_2DLINE	2D Line Chart
CI_FIXED	Constants
CI_LOOPPOS	Loop marker (RealTime)
CI_HORZGRID2	Grid Second Y Axis

Setting:
RGB Color

Data Type
Long

See Also
ItemStyle, ItemWidth

Data

Data

DecimalsNum Property

An integer value (16 bits) that sets or returns the number of decimals used to show specific elements in the chart. The index specifies the item to change the number of decimals.

Visual Basic

```
[form.] Chart1.DecimalsNum(Index) [ = setting% ]
```

Visual C++

```
nDec = pChart1-> GetNumProperty("DecimalsNum",Index);
pChart1-> SetNumProperty("DecimalsNum",nSetting,Index);
```

SQLWindows

```
Set nDec = chart_GetArrNumProp(cc1,"DecimalsNum",Index)
Call chart_SetArrNumProp(cc1,"DecimalsNum",nSetting,Index)
```

Borland C++

```
pChart1->GetPropDecimalsNum(nDec,Index);
pChart1->SetPropDecimalsNum(nSetting,Index);
```

Property Code
CP_DECIMALSNUM

Comments

Constant	Used to change number of decimals used in ...
CD_ALL	All the items.
CD_VALUES	Point Values (Ballon, Dialog and Data-Editor)
CD_YLEG	Y Legend Values.
CD_XLEG	X Legend Values.
CD_YLEG2	Second Y axis Values.

Setting:
Number of decimals

Data Type
Integer

See Also
Decimals

Data

Data

ShowDialog Property

This property is used to show any of the user interface dialogs provided by the library. The index value specifies the dialog to show.

Visual Basic
```
[form.] Chart1.ShowDialog(Index) = setting%
```

Visual C++
```
pChart1-> SetNumProperty("ShowDialog",bSetting,Index);
```

SQLWindows
```
Call chart_SetNumProp(cc1,"ShowDialog",bSetting,Index)
```

Borland C++
```
pChart1->SetPropShowDialog(bSetting,Index);
```

Comments
This function is provided to the end user by the menu or toolbar, if permited by the programmer.

Index	Setting	Meaning
CDIALOG_EXPORTFILE	Not Used (*)	Export File
CDIALOG_IMPORFILE	Not Used (*)	Import File
CDIALOG_WRITETEMPLATE	Not Used (*)	Write
CDIALOG_READTEMPLATE	Not Used (*)	Read
CDIALOG_PAGESETUP	Not Used (*)	Page Setup
CDIALOG_ABOUT	Not Used (*)	About
CDIALOG_OPTIONS	Not Used (*)	Tabbed
CDIALOG_EDITTITLES	Not Used (*)	Edit Titles
CDIALOG_FONTS	Font Types	Font
CDIALOG_ROTATE	Not Used (*)	Rotate
CDIALOG_GENERAL	Not Used (*)	General
CDIALOG_SERIES	Not Used (*)	Series
CDIALOG_SCALE	Not Used (*)	Scale

**** Not used values must be set to zero.***

See Also
How can I display Chart FX internal dialogs without accesing the Toolbar?

Tag Property

Data

This property is commonly used in all Visual Basic Controls, please refer to your VB Help for more information on this topic.

LegendWidth Property

Obsolete API from Chart FX 2.0

This property has been replaced for ToolSize property using the appropiate legend constant. This property used to set the Legend Width.

Data

Data

Scroll Property

This property is used to modify the scrolling position of the chart. The index specifies the scroll code (see wParam in WM_HSCROLL) and the value represents the
scroll additional information (see lParam in WM_HSCROLL).

Visual Basic

```
[form.] Chart1.Scroll(Index) = setting%
```

Visual C++

```
pChart1->SetNumProperty("Scroll",lSetting,Index);
```

SQLWindows

```
Call chart_SetNumProp(cc1,"Scroll",lSetting,Index)
```

Borland C++

```
pChart1->SetPropScroll(;Setting,Index);
```

Property Code
CP_SCROLL

Remarks
You can use this property to "syncronize" two charts in conjunction with the **UserScroll** Event

Data Type
Long

See Also
UserScroll event, Handling Notification Messages

HelpContextId Property

Data

This property is very useful when adding an electronic help file to your application. Basically, Chart FX Control will send you the value contained in this property when the end-user press F1 and Chart control has the focus. This property will allow you to perform context sensitive help files in your application.

This property is commonly used in all Visual Basic Controls, please refer to your Visual Basic documentation for more information on this property.

Data

BarBitmap Property

Data

This property allows you to set a Bitmap handle to create a 2D bar pictogram.

Visual Basic

```
[form.] Chart1.BarBitmap = setting%
```

Visual C++

```
pChart1->SetStrProperty("BarBitmap",sSetting);
```

SQLWindows

```
Call chart_SetStrProp(cc1,"BarBitmap",sSetting)
```

Borland C++

```
pChart1->SetPropBarBitmap(sSetting);
```

Property Code

CP_BARBITMAP

Remarks

The setting must be used in conjunction with the LoadPicture function as follows: (i.e. Visual Basic)

```
Chart1.BarBitmap = LoadPicture(c:\windows\cars.bmp)
```

Data Type

String

MarkerSize Property

This property allows you to change the marker size for all the series in a chart.

Visual Basic

```
[form.] Chart1.MarkerSize = setting%
```

Visual C++

```
pChart1->SetStrProperty("MarkerSize",sSetting);
```

SQLWindows

```
Call chart_SetStrProp(cc1,"MarkerSize",sSetting)
```

Borland C++

```
pChart1->SetPropMarkerSize(sSetting);
```

Property Code
CP_MARKERSIZE

Remarks
Marker Size default value is 3
Marker size must be set between 1 and 20
Important: this message controls the radius of the marker, therefore the size will be doubled.

Data Type
Integer

See Also
MarkerVolume

MarkerVolume Property

This property allows you to set/get proprotion occupied by a marker in its assigned space. This property has effect on x axis for bar charts and z axis for any cluster chart.This message will affect all series.

Visual Basic
```
[form.] Chart1.MarkerVolume = setting%
```

Visual C++
```
pChart1->SetStrProperty("MarkerVolume",sSetting);
```

SQLWindows
```
Call chart_SetStrProp(cc1,"MarkerVolume",sSetting)
```

Borland C++
```
pChart1->SetPropMarkerVolume(sSetting);
```

Property Code
CP_MARKERVOLUME

Remarks
Setting to 100 will have the same effect as CT_TOGETHER.
Setting to 0 will activate the maximum separation.

Data Type
Integer

See Also
MarkerSize, View3DDepth

Data

View3DDepth Property

Data

This property allows you to change or get the marker depth for all the series in a chart.

Visual Basic
```
[form.] Chart1.View3DDepth = setting%
```

Visual C++
```
pChart1->SetStrProperty("View3DDepth",sSetting);
```

SQLWindows
```
Call chart_SetStrProp(cc1,"View3DDepth",sSetting)
```

Borland C++
```
pChart1->SetPropView3DDepth(sSetting);
```

Property Code
CP_VIEW3DDEPTH

Remarks
Default Marker Size value is 100%
Marker depth can be set between 0 and 1000%

100% means the marker will have a depth equals to its width
200% means the marker will have a depth double than its width

Data Type
Integer

See Also
MarkerVolume, MarkerSize

Data

View3DLight Property

Data

This property allows you to activate/deactivate shadow option in drawing 3D charts.

Visual Basic

```
[form.] Chart1.View3DLight = setting%
```

Visual C++

```
pChart1->SetStrProperty("View3DLight",sSetting);
```

SQLWindows

```
Call chart_SetStrProp(cc1,"View3DLight",sSetting)
```

Borland C++

```
pChart1->SetPropView3DLight(sSetting);
```

Property Code
CP_VIEW3DLIGHT

Remarks

Data Type
Boolean

See Also
Color, View3DDepth

Data

Data

Shape Property

This property allows the programmer to create a template file with all the visual attributes (Colors, borders, etc) of the current chart without including data.

```
Visual Basic
[form.] Chart1.Shape = setting$

Visual C++
pChart1->SetStrProperty("Shape",sSetting);

SQLWindows
Call chart_SetStrProp(cc1,"Shape",sSetting)

Borland C++
pChart1->SetPropShape(sSetting);
```

Property Code
CP_SHAPE

Remarks
This property can also be used to change line thickness in 3D Line Charts.To set conic forms the setting should be set to a negative value, otherwise the shape will be cilindric.
Setting this property will affect all series.
To set different shapes to different series please refer to MultiShape property.

Data Type
Integer

See Also
MultiShape, PointType, MultiPoint

MultiType Property

This property is used to set different series to different chart Types. This property can also be used in conjunction with the MultiShape property to give awesome effects to your charts.

Visual Basic
```
[form.] Chart1.MultiType(Index) = setting%
```

Visual C++
```
pChart1-> SetNumProperty("MultiType",bSetting,Index);
```

SQLWindows
```
Call chart_SetNumProp(cc1,"MultiType",bSetting,Index)
```

Borland C++
```
pChart1->SetPropMultiType(bSetting,Index);
```

Property Code
CP_MULTITYPE

Remarks
Chart Types supported as Multi Type: BAR, AREA, CUBE, MARK, LINE, SPLINE and HILOW
In a three-series chart setting a Multiple Type Chart: (VB)
```
Chart1.MultiType(0) = BAR
Chart1.MultiType(1) = CUBE
Chart1.MultiType(2) = SPLINE
```

Data Type
Integer

See Also
MultiShape, MultiPoint, MultiLineStyle, Shape, Chart Types Table

Data

Data

MultiShape Property

This property is used to set different series to different shapes. The only difference with the Shape property is that MultiShape will allow you to set different series by setting an index to the series

Visual Basic

```
[form.] Chart1.MultiShape(Index) = setting%
```

Visual C++

```
pChart1-> SetNumProperty("MultiShape",bSetting,Index);
```

SQLWindows

```
Call chart_SetNumProp(cc1,"MultiShape",bSetting,Index)
```

Borland C++

```
pChart1->SetPropMultiShape(bSetting,Index);
```

Property Code
CP_MULTISHAPE

Remarks
This property can also be used to change line thickness in 3D Line Charts.
To set conic forms the setting should be set to a negative value, otherwise the shape will be cilindric.

Data Type
Integer

See Also
MultiType, MultiLineStyle, MultiPoint, Shape

GalleryTool Property

This property allows you to enable/disable any of the charts available in the Gallery Icon Combo of the Toolbar.

Visual Basic
```
[form.] Chart1.GalleryTool = setting%
```

Visual C++
```
pChart1->SetStrProperty("GalleryTool",sSetting);
```

SQLWindows
```
Call chart_SetStrProp(cc1,"GalleryTool",sSetting)
```

Borland C++
```
pChart1->SetPropGalleryTool(sSetting);
```

Property Code
CP_GALLERYTOOL

Remarks
To enable only Bar and Line Charts: (i.e. Visual Basic)
```
Chart1.GalleryTool = CSG_BAR Or CSG_LINE
```

Data Type
Long

See Also
CustomTool, Customizing the Toolbar

ClearLegend Property

This property allows you to clear an specific legend in the chart..

```
Visual Basic
[form.] Chart1.ClearLegend(Index) = setting%
```

```
Visual C++
pChart1->SetStrProperty("ClearLegend",sSetting,Index);
```

```
SQLWindows
Call chart_SetStrProp(cc1,"ClearLegend",sSetting,Index)
```

```
Borland C++
pChart1->SetPropClearLegend (sSetting, Index);
```

Property Code
CP_CLEARLEGEND

Comments

CHART_LEGEND	Clear Point Legend
CHART_SERLEG	Clear Series Legend
CHART_KEYLEG	Clear Key Legend (x-axis)
CHART_KEYSER	Clear Key Series Legend
CHART_FIXLEG	Clear constant Legend.
CHART_YLEG	Clear Y axis Legend.

Data Type
Integer

See Also
Legend, SerLeg, KeyLeg, KeySer, Customizing Y Axis Legend

MaxValues Property

This property allows you to set the buffersize for a Limited RealTime chart. This message must be called before the OpenData property.

Visual Basic

```
[form.] Chart1.MaxValues = setting%
```

Visual C++

```
pChart1->SetStrProperty("MaxValues",sSetting);
```

SQLWindows

```
Call chart_SetStrProp(cc1,"MaxValues",sSetting)
```

Borland C++

```
pChart1->SetPropMaxValues(sSetting);
```

Property Code
CP_MAXVALUES

Remarks
Important Note: Setting the BufferSize will reset the values, this is why you have to make the call before the OpenData statement.

Data Type
Integer

See Also
RealTime Charts

Data

RealTimeStyle Property

Data

This property allows you to set/get the Real Time style of a chart.

Visual Basic
```
[form.] Chart1.RealTimeStyle = setting%
```

Visual C++
```
pChart1->SetStrProperty("RealTimeStyle",sSetting);
```

SQLWindows
```
Call chart_SetStrProp(cc1,"RealTimeStyle",sSetting)
```

Borland C++
```
pChart1->SetPropRealTimeStyle(sSetting);
```

Property Code
CP_REALTIMESTYLE

Remarks
Setting:

CRT_LOOPPOS	Show Loop Marker
CRT_NOWAITARROW	Hide HourGlass cursor

Data Type
Long

See Also
RealTime Charts, MaxValues

Export Property

This property allows the programmer to export data and chart using different formats. Also used to save chart as a WMF.

Visual Basic
```
[form.] Chart1.Export(Index) = setting%
```

Visual C++
```
pChart1->SetNumProperty("Export,Index,setting%)
```

SQLWindows
```
Call chart_SetNumProp(cc1,"Export",Index,setting%)
```

Borland C++
```
pChart1->SetPropExport(Index,setting%);
```

Property Code
CP_EXPORT

Remarks
Index: index of the datatype

Remarks
Index Contains an index of the data type to be exported (See Comments)

Setting contains:	NULL = interact with the clipboard or Handle to a File

Return Value
None (must not be used)

Comments
Index can be set to any of the following values:

CHART_DATA	copy the data using a Tab Separated Values format (TSV)
CHART_BITMAP	copy the chart picture as a Windows Bitmap format
CHART_METAFILE	copy the chart picture as a Windows Metafile format
CHART_INTERNALFILE	Saves the chart receiving in IParam a pointer to a file that must opened and ready to receive information.
CHART_INTERNALTEMPLATE	Saves the template of the chart receiving in IParam a pointer to a file that must opened and ready to receive the information

Data Type
Long

See Also
ExportStr, Import, ImportStr

TbBitmap Property

This property allows you to set a new Bitmap as the toolbar picture.

Visual Basic
```
[form.] Chart1.TbBitmap = setting%
```

Visual C++
```
pChart1->SetStrProperty("TbBitmap",sSetting);
```

SQLWindows
```
Call chart_SetStrProp(cc1,"TbBitmap",sSetting)
```

Borland C++
```
pChart1->SetPropTbBitmap(sSetting);
```

Property Code
CP_TBBITMAP

Remarks
The setting must be used in conjunction with the LoadPicture function as follows: (i.e. Visual Basic)
```
Chart1.TbBitmap = LoadPicture(c:\windows\newtool.bmp)
```

Data Type
String

See Also
TbItemId, TbItemStyle, EnableTbItem, CustomTool, Customizing the Toolbar

TbItemId Property

This property is used to set/get the ID for any of the items in the Toolbar. For more information on how to use this property please refer to Customizing the Toolbar

Visual Basic

```
[form.] Chart1.TbItemID(Index) = setting%
```

Visual C++

```
pChart1-> SetNumProperty("TbItemID",bSetting,Index);
```

SQLWindows

```
Call chart_SetNumProp(cc1,"TbItemID",bSetting,Index)
```

Borland C++

```
pChart1->SetPropTbItemID(bSetting,Index);
```

Property Code
CP_TBITEMID

Remarks
This message is only used when customizing the Toolbar.
Item Index is zero based with the most left item of the Toolbar as zero.

Sample:
Please refer to Customizing the Toolbar

Data Type
Long

See Also
TbBitmap, TbItemStyle, EnableTbItem, CustomTool, Customizing the Toolbar

Data

Data

TbItemStyle Property

This property is used to set/get the style for any of the items in the Toolbar. For more information on how to use this property please refer to Customizing the Toolbar

Visual Basic
```
[form.] Chart1.TbItemStyle(Index) = setting%
```

Visual C++
```
pChart1-> SetNumProperty("TbItemStyle",bSetting,Index);
```

SQLWindows
```
Call chart_SetNumProp(cc1,"TbItemStyle",bSetting,Index)
```

Borland C++
```
pChart1->SetPropTbItemStyle(bSetting,Index);
```

Property Code
CP_TBITEMSTYLE

Remarks
This message is only used when customizing the Toolbar.
Item Index is zero based with the most left item of the Toolbar as zero.

Sample:
Please refer to Customizing the Toolbar

Data Type
Long

See Also
TbBitmap, TbItemId, EnableTbItem, CustomTool, Customizing the Toolbar

Data

Data

EnableTbItem Property

This property is used to enable/disable any of the Toolbar Items.

Visual Basic
```
[form.] Chart1.EnableTbItem(Index) = setting%
```

Visual C++
```
pChart1-> SetNumProperty("EnableTbItem",bSetting,Index);
```

SQLWindows
```
Call chart_SetNumProp(cc1,"EnableTbItem",bSetting,Index)
```

Borland C++
```
pChart1->SetPropEnableTbItem(bSetting,Index);
```

Property Code
CP_ENABLETBITEM

Remarks
0 setting will disbale the Toolbar Item
1 setting will enable the Toolbar Item

Data Type
Boolean

See Also
TbBitmap, TbItemId, TbItemStyle, CustomTool, Customizing the Toolbar

MultiYAxis Property

This property is used to activate double Y axis in the chart, and assign different series to this secondary y axis.

Visual Basic

```
[form.] Chart1.MultiYAxis(Index) = setting%
```

Visual C++

```
pChart1-> SetNumProperty("MultiYAxis",bSetting,Index);
```

SQLWindows

```
Call chart_SetNumProp(cc1,"MultiYAxis",bSetting,Index)
```

Borland C++

```
pChart1->SetPropMultiYAxis(bSetting,Index);
```

Property Code
CP_MULTIYAXIS

Remarks
Please refer to Adm property for controling secondary y axis settings.

Sample:
To assign secondary axis to the third series of a chart (i.e. VB)

```
Chart1.MultiYAxis(2) = 1
```

Data Type
Integer

See Also
Adm Property,

ToolStyle Property

This property is used to set the styles of the Tools in the Chart Window (Toolbar and Legends).

Visual Basic
```
[form.] Chart1.ToolStyle(Index) = setting%
```

Visual C++
```
pChart1-> SetNumProperty("ToolStyle",bSetting,Index);
```

SQLWindows
```
Call chart_SetNumProp(cc1,"ToolStyle",bSetting,Index)
```

Borland C++
```
pChart1->SetPropToolStyle(bSetting,Index);
```

Property Code
CP_TOOLSTYLE

Remarks
Available Tools (Index):

CTOOL_TB	Apply to Toolbar
CTOOL_LEGEND	Apply to Values Legend
CTOOL_SERLEGEND	Apply to Series Legend
CTOOL_OPTIONS	Access Internal Options (Must be use in combination)

Setting (Combination of):

CTS_HIDEFOCUS	Toolbar hide when not active
CTS_WHITELINE	Draw Shadow white line
CTS_DELIMITER	Draw delimiter when child
CTS_SIZEABLE	Sizeable when child
CTS_HORZLAYER	Layerable when Horizontal
CTS_VERTLAYER	Layerable when Vertical
CTS_SIZELAYER	Sizeable when layered
CTS_DBLCLKS	Accept Dblclcks - To (un)dock
CTS_DOCKABLE	Dockable
CTS_SPLITTER	Draw Splitter
CTS_3DFRAME	Draw 3D Frame
CTS_BORDERLAYER	LayerBorder always
CTS_BORDERIFLAYER	LayerBorders when layered

Data Type
Long

When using the CTOOL_OPTIONS combined with any of the tools in the wParam, for example
```
Chart1.ToolStyle(CTOOL_TB Or CTOOL_OPTIONS) = setting%
```

You may combine the following constants in the lParam:

Options with Toolbar (CTOOL_TB)

CHART_TBBALLOON	Balloon ToolTips
CHART_TBSTANDARD	Standard ToolTips

CHART_TBNOTOOLTIPS	No ToolTips in the Toolbar

Options with Series and Point Legends (CTOOL_LEGEND or CTOOL_SERLEGEND)

CHART_LWORDBREAK	Multiline Legends
CHART_LSKIPEMPTY	Do not show empty legends
CHART_LSHOWMENU	Show menu on Demand
CHART_LOPTIONSDLG	Show Internal Options Dlg.
CHART_LRIGHTALIGN	Align legends to Right.

Samples:

To make series legend sizeable with 3D border:

```
Chart1.ToolStyle(CTOOL_SERLEGEND) = CTS_SIZEABLE Or CTS_3DFRAME
```

To align points legend to the right:

```
Chart1.ToolStyle(CTOOL_LEGEND Or CTOOL_OPTIONS) = CHART_LRIGHTALIGN
```

To deactivate ToolTips programatically:

```
Chart1.ToolStyle(CTOOL_TB Or CTOOL_OPTIONS) = CHART_TBNOTOOLTIPS
```

See Also

ToolPos, ToolSize

Data

Zoom Property

Data

This property allows you to Turn on/off the Zoom mode. This property is not intended to perform a Zoom programatically its function is to simulate Zoom button pressing/depressing.

Visual Basic
```
[form.] Chart1.Zoom = setting%
```

Visual C++
```
pChart1->SetStrProperty("Zoom",sSetting);
```

SQLWindows
```
Call chart_SetStrProp(cc1,"Zoom",sSetting)
```

Borland C++
```
pChart1->SetPropZoom(sSetting);
```

Property Code
CP_ZOOM

Remarks
Turning on the Zoom mode will allow your end users to select a region in the chart to zoom in.

Data Type
Integer

Data

ItemBkColor Property

Data

A Color (Long) value that sets or returns the background color used to paint the item specified in the index.

Visual Basic

```
[form.] Chart1.ItemBkColor(Index) [ = setting& ]
```

Visual C++

```
lColor = pChart1-> GetNumProperty("ItemBkColor",Index);
pChart1-> SetNumProperty("ItemBkColor",lSetting,Index);
```

SQLWindows

```
Set lColor = chart_GetArrNumProp(cc1,"ItemBkColor",Index)
Call chart_SetArrNumProp(cc1,"ItemBkColor",lSetting,Index)
```

Borland C++

```
pChart1->GetPropItemBkColor(lColor,Index);
pChart1->SetPropItemBkColor(lSetting,Index);
```

Property Code

CP_ITEMBKCOLOR

Comments

CI_HORZGRID	Horizontal grid
CI_VERTGRID	Vertical grid
CI_2DLINE	2D Line Chart
CI_FIXED	Constants
CI_LOOPPOS	Loop marker (RealTime)
CI_HORZGRID2	Grid Second Y Axis

Setting:

RGB Color

Data Type

Long

See Also

ItemColor, ItemStyle, ItemWidth

Data

Import Property

Data

This property allows the programmer to retrieve files saved with the Export Property as a handle to a file.

Visual Basic

```
[form.] Chart1.Import(Index) [ = setting$ ]
```

Visual C++

```
pChart1->SetStrProperty("Import",sSetting,Index);
```

SQLWindows

```
Call chart_SetArrStrProp(cc1,"Import",sSetting,Index)
```

Borland C++

```
pChart1->SetPropImport(sSetting,Index);
```

Property Code
CP_IMPORT

Index:
CHART_INTERNALFILE
CHART_INTERNALTEMPLATE

Setting:
Handle to a file.

Data Type
Long

See Also:
ImportStr, Export, ExportStr

SeparateSlice Property

This property is used to separate an specific slice from a doughnut or pie chart at run time.

Visual Basic

```
[form.] Chart1.SeparateSlice(Index) = setting%
```

Visual C++

```
pChart1-> SetNumProperty("SeparateSlice",bSetting,Index);
```

SQLWindows

```
Call chart_SetNumProp(cc1,"SeparateSlice",bSetting,Index)
```

Borland C++

```
pChart1->SetPropSeparateSlice(bSetting,Index);
```

Property Code
CP_SEPARATESLICE

Remarks
Index:
Contains the point that represents the slice to be separated.

Setting:
Contains the separation distance measured from the center of the pie expressed in radius percentage

Data Type
Integer

See Also
StyleEx

Data

PaintInfo Property

Data

Please refer to "Customizing Chart Painting" for more information on this property.

Data

OptionsDlg

Data

This property will allow you to pop-up Chart FX Tabbed dialog to pre-set different settings in the Chart Control. **Please refer to your Chart FX manual for all options located in this Tabbed dialog**.

ToolSize Property

This property is used to set/get size any of the legends of the chart.

Visual Basic

```
[form.] Chart1.ToolSize(Index) = setting%
```

Visual C++

```
pChart1-> SetNumProperty("ToolSize",bSetting,Index);
```

SQLWindows

```
Call chart_SetNumProp(cc1,"ToolSize",bSetting,Index)
```

Borland C++

```
pChart1->SetPropToolSize(bSetting,Index);
```

Property Code
CP_TOOLSIZE

Remarks
Index can be set to:

CTOOL_LEGEND	Points Legend
CTOOL_SERLEGEND	Series Legend

If Legend is Right or Left Aligned Height does not apply.
If Legend is Top or Bottom Aligned Width does not apply.
If Legend is floatable both Width and Height apply

Sample:
To modify the Width the series legend when Right Aligned:

```
Chart1.ToolSize(CTOOL_SERLEGEND) = CHART_ML(100,0)
```

Data Type
Long

See Also
ToolStyle, ToolPos

ExportStr Property

This property allows the programmer to export data and chart using different formats and setting is a string, For numeric settings please refer to Expor property. Also used to save chart as a WMF.

Visual Basic

```
[form.] Chart1.ExportStr(Index) = setting%
```

Visual C++

```
pChart1->SetNumProperty("ExportStr,Index,setting%)
```

SQLWindows

```
Call chart_SetNumProp(cc1,"ExportStr",Index,setting%)
```

Borland C++

```
pChart1->SetPropExportStr(Index,setting%);
```

Property Code
CP_EXPORTSTR

Remarks
Index Contains an index of the data type to be exported (See Comments)

Setting contains: LPSTR containing the file (including path)

Return Value
None (must not be used)

Comments
Index can be set to any of the following values:

CHART_METAFILE	copy the chart picture as a Windows Metafile format
CHART_CFXFILE	Saves a chart (using propietary format) to a file
CHART_CFXTEMPLATE	Saves a Template (using propietary format)

Data Type
String

See Also:
Export, Import, ImportStr

Data

ImportStr Property

Data

This property allows the programmer to retrieve files saved with the ExporStr Property.

Visual Basic

```
[form.] Chart1.Import(Index) [ = setting$ ]
```

Visual C++

```
pChart1->SetStrProperty("Import",sSetting,Index);
```

SQLWindows

```
Call chart_SetArrStrProp(cc1,"Import",sSetting,Index)
```

Borland C++

```
pChart1->SetPropImport(sSetting,Index);
```

Property Code
CP_IMPORTSTR

Index:
CHART_CFXFILE
CHART_INTERNALFILE

Setting:
String including path and name of the chart to retrieve.

Data Type
String

See Also
Import, Export, ExportStr

DataSource Property

This property is used when having charts linked to a data control (DataBound Charts), please refer to DataBound Charts, for more information on how to link a chart control to a database.

See Also

DataBound, DataStyle Property, DataType property.

DataType Property

This property is used to specify field attributes in a databound chart. With this property you can alter the default behavior that Chart FX applies when linked to a database.

Visual Basic
```
[form.] Chart1.DataType(Index) = setting%
```

Visual C++
```
pChart1-> SetNumProperty("DataType",bSetting,Index);
```

SQLWindows
```
Call chart_SetNumProp(cc1,"DataType",bSetting,Index)
```

Borland C++
```
pChart1->SetPropDataType(bSetting,Index);
```

Property Code
CP_DATATYPE

Remarks
DataType Index is the field index in the SELECT statement (Zero Based)

DataType setting can be any of the following:

CDT_STRING	specify a string field type
CDT_NUMBER	specify a value field type
CDT_NOTUSED	do not use that field to plot in the chart.

Sample:
The default behavior is that Chart FX will plot the year as another series since it is a number field and therefore it will be placed in the chart. Now, if the chart you want to make is one with the x axis containing the year and plot the sales and projected sales in a different series without using the return and name fields you will fill the DataType array as follows:

```
 1st we have to convert year field in a string to be selected as a x axis legend.
Chart1.DataType(0)=CDT_STRING
Then assigned the CDT_NUMBER constant to the number fields
Chart1.DataType(1)  = CDT_NUMBER
Chart1.DataType(2)  = CDT_NUMBER
Finally, assign CDT_NOTUSED to those fields we dont want to plot.
Chart1.DataType(3)  = CDT_NOTUSED
Chart1.DataType(4)  = CDT_NOTUSED
```

Important Name: when using the DataType property, you must specify the attribute for all fields in the select statement.

Data Type
Integer

See Also
DataStyle, DataBound, DataSource

Data DataStyle Property

An integer property that changes the default behavior for databound charts, with this property you can specify how Chart FX will plot the fields in the SELECT statement bound to the chart control. Since Chart FX applies default rules to construct databound charts in terms how the fields are take to pass legends and pints, with this property you can change these default rules.

Visual Basic
```
[form.] Chart1.DataStyle [ = setting! ]
```

Visual C++
```
pChart1->SetFloatProperty("DataStyle",fSetting);
```

SQLWindows
```
Call chart_SetNumProp(cc1,"DataStyle",fSetting)
```

Borland C++
```
pChart1->SetPropDataStyle(fSetting);
```

Property Code
CP_DATASTYLE

Remarks
DataStyle setting can be a combination of the following:

CHART_DS_SERLEGEND	Take field names as series legends. Default = ON
CHART_DS_USEDATEASLEG	Use date fields as legends. Default = OFF
CHART_DS_USETEXTASLEG	Use text fields as legends. Default = ON

The default rules that can be changed with the DataStyle properties are:

Chart FX will apply default rules to construct the chart when linked to a Data control. These rules are somehow intelligent in picking the information from the database and assign the legends to it, so if you send a SELECT statement, Chart FX will create the chart series and point legends automatically. These rules are:
1) Series Legends will be taken from the numerical field names

2) All numerical columns will be plotted as different series and all string and/or date columns will be plotted as point legends (joined by the - character).

3) All string and numerical fields specified in the SELECT statement will be plot.

Data Type
Integer

See Also
DataType, DataBound, DataSource

Reserved4 Property

Data

Reserved property used for compatibility among different development tools. ***Do not use!***

ToolPos Property

This property is used to set a position for a specific tool (Toolbar or Legends) in the chart window

Visual Basic
```
[form.] Chart1.ToolPos(Index) = setting%
```

Visual C++
```
pChart1-> SetNumProperty("ToolPos",bSetting,Index);
```

SQLWindows
```
Call chart_SetNumProp(cc1,"ToolPos",bSetting,Index)
```

Borland C++
```
pChart1->SetPropToolPos(bSetting,Index);
```

Property Code
CP_TOOLPOS

Remarks
Available Tools (Index):
CTOOL_TB, CTOOL_LEGEND, CTOOL_SERLEGEND, CTOOL_MOVE.

Comments
wParam can be set to any of the following:

CTOOL_TB	Apply position to the Toolbar
CTOOL_LEGEND	Apply position to values Legend
CTOOL_SERLEGEND	Apply position to Series Legend
CTOOL_MOVE	To move the tool when floating(must be use in combination) lParam change its meaning

lParam can be set to any of the following:

CTP_TOP	Align Tool to top
CTP_LEFT	Align Tool to left
CTP_BOTTOM	Align tool to bottom
CTP_RIGHT	Align tool to right
CTP_FIXED	Tool at fixed position (Only Legends)
CTP_FLOAT	Make Tool floatable
CTP_SWITCH	Switch between floating and child

If CTOOL_MOVE is used in combination with any of the tools setting must be:
CHART_ML(Left, Right) relative to the chart Window.

Data Type
Integer

See Also
ToolStyle, ToolSize

Data

Data

TypeEx Property

This property allows you to set/get additional chart types or general settings to the chart.

Visual Basic
```
[form.] Chart1.TypeEx = setting%
```

Visual C++
```
pChart1->SetStrProperty("TypeEx",sSetting);
```

SQLWindows
```
Call chart_SetStrProp(cc1,"TypeEx",sSetting)
```

Borland C++
```
pChart1->SetPropTypeEx(sSetting);
```

Property Code
CP_TYPEEX

Remarks
Setting is a logical combination of CTE_* constants.

Comments

CTE_STEPLINES	Convert a Line chart to step lines
CTE_SMOOTH	Apply a BitBlitz technique when repainting the chart (Always).
CTE_SQUAREPIE	Force pie charts to be contained in a square (no matter the window size).
CTE_NOLEGINVALIDATE	Useful for RealTime charts.
CTE_ACTMINMAX	Recalculate Min-Max when changing data.
CTE_NOTITLESHADOW	Turn off 3D effect of Top Title.
CTE_CREATELEGENDS	Allows the end-users to create legends from Data Editor.
CTE_NOCROSS	Turn off cross-hairs feature.
CTE_LOGBREAK	Break logarithmic scale every Log Base nth points, for every break, y gap is multiplied byLog Base. This feature is used to prevent grid lines all appear in top of the chart.

Data Type
Long

See Also
Type, Style, StyleEX, CloseData (for COD_SMOOTH)

StyleEx Property

This property allows you to set/get additional chart styles or general settings to the chart.

Visual Basic

```
[form.] Chart1.StyleEx = setting%
```

Visual C++

```
pChart1->SetStrProperty("StyleEx",sSetting);
```

SQLWindows

```
Call chart_SetStrProp(cc1,"StyleEx",sSetting)
```

Borland C++

```
pChart1->SetPropStyleEx(sSetting);
```

Property Code
CP_STYLEEX

Remarks
Setting is a logical combination of CSE_* constants.

Comments
lParam can be a logical OR of the following constants:
CSE_NOSEPARATE Prohibit end users to separate slices from a pie or a doughnut chart.

Data Type
Long

See Also
Style, Type, TypeEx

MouseCapture Property

This property allows you to set/get additional chart styles or general settings to the chart.

Visual Basic

```
[form.] Chart1.MouseCapture = setting%
```

Visual C++

```
pChart1->SetStrProperty("MouseCapture",sSetting);
```

SQLWindows

```
Call chart_SetStrProp(cc1,"MouseCapture",sSetting)
```

Borland C++

```
pChart1->SetPropMouseCapture(sSetting);
```

Property Code
CP_MOUSECAPTURE

Remarks
Setting
TRUE = capture.
FALSE = release.
When capturing the mouse (for mouse tracking features) it is imperative that you release it again. Normally you will call this message in any of the ButtonDown events and release it again in the ButtonUp event.

Data Type
Boolean

See Also
How do I capture mouse to drag a point to a desired location?
Handling Notification Messages

LegStyle Property

An Integer value (16 bits) that sets or returns the type of the legend style. This property affects how this legend is presented in a chart.

Visual Basic

```
[form.] Chart1.LegStyle [ = setting% ]
```

Visual C++

```
nWidth = pChart1->GetNumProperty("LegStyle");
pChart1->SetNumProperty("LegStyle",nSetting);
```

SQLWindows

```
Set nWidth = chart_GetNumProp(cc1,"LegStyle")
Call chart_SetNumProp(cc1,"LegStyle",nSetting)
```

Borland C++

```
pChart1->GetPropLegStyle(nWidth);
pChart1->SetPropLegStyle(nSetting);
```

Remarks
The value must be one of the following flags:

Comments

CL_NOTCLIPPED	Do not clip the X legends (Its programmers responsability to assure that the legends dont overlap each other).
CL_NOTCHANGECOLOR	Do not change the color of the legends that dont fit in the available space (the default behavior is to draw that legends in RED)
CL_HIDEXLEG	Do not draw the X axis Legend.
CL_2LEVELS	Display X axis Labels Up-Down
CL_HIDEYLEG	Do not draw the Y axis Legend.
CL_VERTXLEG	Vertical X Axis labeling. True Type Only
CL_SHOWZLEG	Display Series Legend in Z-axis
CL_GETLEGEND	Send a message every time it needs to draw a legend in the Y axis. You can use the **CM_GETHTEXT** and **CM_SETHTEXT** to get and set the string to draw. This message (**CN_GETLEGEND**) will also be sent for the X axis in a scatter chart.

Data Type
Integer

See Also
ToolStyle, ToolPos, ToolSize

MultiLineStyle Property

This property is used to set/get different line settings for a 2D Line chart. This message will allow you to have different series in different line styles

Visual Basic

```
[form.] Chart1.MultiLineStyle(Index) = setting%
```

Visual C++

```
pChart1-> SetNumProperty("MultiLineStyle",bSetting,Index);
```

SQLWindows

```
Call chart_SetNumProp(cc1,"MultiLineStyle",bSetting,Index)
```

Borland C++

```
pChart1->SetPropMultiLineStyle(bSetting,Index);
```

Property Code
CP_MULTILINESTYLE

Remarks
When setting different styles (other than solid) the width must be 1.
Setting Index to -1 will erase current settings.

Sample:
In a 2D line chart to set two series to different colors:

```
Chart1.MultiLineStyle(0) = CHART_ML(1,CHART_DASH)
Chart1.MultiLineStyle(1) = CHART_ML(4,CHART_SOLID)
```

Data Type
Integer

See Also
ItemColor, ItemWidth, ItemStyle

Data

CurrentAxis Property

Data

This property allows you to set/get additional chart styles or general settings to the chart.

Visual Basic

```
[form.] Chart1.CurrentAxis = setting%
```

Visual C++

```
pChart1->SetStrProperty("CurrentAxis",sSetting);
```

SQLWindows

```
Call chart_SetStrProp(cc1,"CurrentAxis",sSetting)
```

Borland C++

```
pChart1->SetPropCurrentAxis(sSetting);
```

Property Code

CP_CURRENTAXIS

Remarks

Setting

0 = Primary Y axis
1 = Secondary Y axis
2 = X axis (Scatter)

Data Type

Integer

See Also

Customizing Chart Painting

Data

Enabled Property

Data

Common property used in all controls that allows you to enable/disable the chart control inside your form.

Data

CustomTool Property

Data

A Long value (32 bits) that sets or returns the visible buttons in the ToolBar.

Visual Basic

```
[form.] Chart1.CustomTool [ = setting& ]
```

Visual C++

```
lTool = pChart1->GetNumProperty("CustomTool");
pChart1->SetNumProperty("CustomTool",lSetting);
```

SQLWindows

```
Set lTool = chart_GetNumProp(cc1,"CustomTool")
Call chart_SetNumProp(cc1,"CustomTool",lSetting)
```

Borland C++

```
pChart1->GetPropCustomTool(lTool);
pChart1->SetPropCustomTool(lSetting);
```

Property Code

CP_CUSTOMTOOL

Remarks

Setting of this property must contain a Bitwise OR of Tool Constants located in your include file

Data Type

Long

See Also

GalleryTool, Customizing the Toolbar

MultiPoint Property

This property is used to set/get assign different point settings in a chart that contains points (LIN, SPLINE, POLAR, etc). This message will allow you to have different series in different point styles

Visual Basic

```
[form.] Chart1.MultiPoint(Index) = setting%
```

Visual C++

```
pChart1-> SetNumProperty("MultiPoint",bSetting,Index);
```

SQLWindows

```
Call chart_SetNumProp(cc1,"MultiPoint",bSetting,Index)
```

Borland C++

```
pChart1->SetPropMultiPoint(bSetting,Index);
```

Property Code
CP_MULTIPOINT

Remarks
Index = Series Index or -1 to clear array
Setting = Pre-defined point style or Negative Value of an ASCII
If you want to assign specific point types (Wingdings symbol) you can specify in the lParam the negative values of the correspondent ASCII value in the char map table. The default font is Wingdings. (You have to modify CHARTFX2.INI to change this default font).

Sample:

To set the ½ wingdings symbol to the first series:

```
Chart1.MultiPoint(0) = -171
```

Data Type
Integer

See Also
PointType, MultiShape, MultiType, MultiLineStyle

chart_Get

double chart_Get(HWND *hChart,* **long** *(nSerie,nPoint),* **UINT** *wCode***)**

chart_Get is a function that allows the programmer to obtain the specific value for serie-point of the chart.

Parameter	Name	Description
HWND	hWnd	Chart handle returned by chart_Create function
LONG	nSerie, nPoint	Series and Point of the chart
UINT	wCode	Code of value to get (See comments)

Return Value
Double value

Comments

CHART_GVALUES	Get Y values
CHART_GXVALUES	Get X values (***for scatter charts)***
CHART_GINIVALUES	Get Initial values (***for bar charts***)

Sample
To get the value of the third point, first series of a chart:

```
d = chart_Get(Chart1.hWnd,CHART_ML(0,3),CHART_GVALUES);
```

Data chart_Get2

N/A

chart_GetAdm

Data

chart_SetStatusItem

long chart_SetStatusItem(HWND *hChart,* **int** *nIndex,* **BOOL** *bHaveText,* **UINT** *ID,* **BOOL** *bFramed,* **int** *nWidth,* **int** *nMin,* **int** *nDesp,* **DWORD** *dwStyle***)**

Some development tools do not support CHART_STITEMSTRUCT pointer passing (i.e. Visual Basic), Therefore you must use the chart_SetStatusItem function to pass the different items that are going to appear in the status bar.

```
Visual Basic
Use chart_SetStatusItem function (DLL model)

Visual C++
Use Status property (Please refer to properties)

SQLWindows
Use chart_SetStatusItem function (DLL model)

Borland C++
Use Status property (Please refer to properties)
```

Remember that in order to access the chart_SetStatusItem function (i.e. from VB) you must first obtain the chart window handle by doing the following:

```
Microsoft Visual Basic
hWnd = Chart1.hWnd

Gupta SQLWindows 4.1
* SalVBXGetProp(cc1, sValue, 'hCtlWnd', 0)
* Set hWnd = SalStrToNumber(sValue)
```

After obtaining the window handle you may use the chart_SetStatusItem function to interact with the chart.

See Also
ShowStatus, How do I create a status bar?

chart_SetStripe

long chart_SetStripe(HWND *hChart,* **int** *nIndex,* **double** *fBegin,* **double** *fEnd,* **DWORD** *dwColor***)**

chart_SetStripe is a function that allows the programmer to pass two values in which a color frame is to be displayed in the background of the chart. This function is very useful when you want to denote a specific area in the chart. This function must be accessed after calling the chart_OpenData function with the COD_STRIPES constant.

Parameter	**Name**	**Description**
HWND	hWnd	Chart handle returned by chart_Create function
int	nIndex	Index of the stripe
double	fBegin	Numeric value corresponding to the beginning of the stripe
double	fEnd	Numeric value corresponding to the ending of the stripe
DWORD	dwColor	RGB color of the stripe

Return Value
See return code of data functions.

Comments
This function has no effect without a OpenData property
This function can apply to any type of charts (except pie charts).
OLE Control (OCX) supports Color Stripe Setting at design time.

See Also
OpenData , What is the use for color stripes and constant lines?

chart_Paint

LONG chart_Paint(HWND *hChart,* **HDC** *hDC,* **int** *nLeft,* **int** *nTop,* **int** *nRight,* **int** *nBottom,* ***UINT*** *wAction,*
LONG *lReserved***)**

chart_Paint is a function that allows the programmer to draw a chart in any device context. This function is very useful when you want to print charts and others objects in the same page or more than one chart in a page.

Parameter	**Name**	**Description**
HWND	hWnd	Chart handle returned by chart_Create function
HDC	hDC	Device context where the chart is going to be drawn.
int	nLeft	x-coordinate of the upper-left corner of the bounding rectangle.
int	nTop	y-coordinate of the upper-left corner of the bounding rectangle.
int	nRight	x-coordinate of the bottom-right corner of the bounding rectangle.
int	nBottom	y-coordinate of the bottom-right corner of the bounding rectangle.
UINT	wAction	**CPAINT_BKGND** if painting background **CPAINT_PRINT** is the chart is to be printed
LONG	lReserved	Reserved. Must be set to 0.

Comments
The bounding rectangle must be in device units.

See Also
How Do I print several charts in the same page? , PrintIt

chart_GetPaintInfo

LONG chart_SetXValue(HWND *hChart,* **int** *nIndex,* **LONG** *lParam***)**

chart_GetPaintInfo is a function that is used when obtaining pertinent information when customizing chart painting and is required to pass a pointer in the lParam. Some development tools (i.e. Visual Basic) does not support casting. Therefore this function is available depending on the information you want to receive.

Parameter	Name	Description
HWND	hWnd	Chart handle returned by chart_Create function
int	nIndex	CPI constant
LONG	lParam	Value depending on the CPI constant passed.

Comments
Please refer to Obtaining pertinent information when customizing chart painting table in page 94.

See Also
Customizing chart Painting