



DFL Software

## Light Lib Business 1.0

Languages	<a href="#"><u>CA-Visual Objects</u></a> <a href="#"><u>MS-Visual Basic</u></a> <a href="#"><u>C/C++</u></a>
DLL Support	<a href="#"><u>DLL Functions</u></a> <a href="#"><u>Callback Functions</u></a> <a href="#"><u>Constants</u></a>
General	<a href="#"><u>Introduction</u></a> <a href="#"><u>Quick Start</u></a> <a href="#"><u>How to use this Help</u></a> <a href="#"><u>Overview</u></a> <a href="#"><u>Compatibility</u></a>
Appendices	<a href="#"><u>Common Problems/Questions</u></a> <a href="#"><u>Editions</u></a> <a href="#"><u>Light Lib Products...</u></a> <a href="#"><u>Technical Support</u></a> <a href="#"><u>License</u></a> <a href="#"><u>About DFL...</u></a>

## DLL Functions

We strongly suggest that you use the extensive support classes and functions provided for the individual languages instead of calling the DLLs directly. The following DLL functions are provided for reference and should not be called directly unless you have a thorough understanding of how to use them.

### Light Lib Business

[bMove\(\)](#)  
[bOnMouse\(\)](#)  
[bPrint\(\)](#)  
[bStabilize\(\)](#)

### Light Lib Objects

[oAccess\(\)](#)  
[oAssign\(\)](#)  
[oNew\(\)](#)  
[oDel\(\)](#)

## Callback Functions

Light Lib Business uses "Callback" functions to allow the DLL to call a user defined function at specific execution points. For example, when a mouse event (eg. left click) occurs on a graph, the DLL in turn calls a standard mouse handling function to process the mouse event. This mouse handler is a Callback function.

## Referencing objects from a Callback Function

In general, graphs are attached to objects (for example a window), but when Light Lib Business executes a callback function which is not a method, the reference to "self" is lost. In this case, we suggest that "self" be passed in one of the user defined parameters. This gives your callback function a reference to "self" which provides access to all the methods and instance variables.

## OnMouse Codeblocks

Each Light Lib Business object has an OnMouse codeblock which receives the following parameters when a mouse event occurs.

### Arguments

*oLLBusiness* Any Light Lib Business object. (eg Graph, Legend, Column etc.)

*oLLSubObject* Any Light Lib Business sub-object if applicable.

*r8Data* A value if applicable.

*oWindow* A reference to the Window object containing the Light Lib Business object.

*oMouseEvent* The mouse event which occurred.

### Description

If the mouse event happens on a graph object or on any of the graph's sub-objects, the OnMouse codeblock associated with the selected object will be evaluated with the parameters listed above.

### Example

See [ExecOnMouse\(\)](#) method from GraphInWindow Class

## Constants

These constants are to be used in conjunction with direct [DLL function calls](#).

### General

[Axis](#)

[Column](#)

[Fonts](#)

[Graph](#)

[Legend](#)

[Error](#)

[System](#)

[Light Lib Objects Constants](#)

## **System Constants**

LLB\_CLASS\_APPLICATION  
LLB\_CLASS\_COLUMN  
LLB\_CLASS\_FONT  
LLB\_CLASS\_GRAPH  
LLB\_CLASS\_LEGEND  
LLB\_CLASS\_X\_AXIS  
LLB\_CLASS\_Y\_AXIS

## Legend

LLB\_LEGEND\_ON\_BEST  
LLB\_LEGEND\_ON\_BOTTOM  
LLB\_LEGEND\_ON\_LEFT  
LLB\_LEGEND\_ON\_MOUSE     Pointer to a callback function  
LLB\_LEGEND\_ON\_RIGHT  
LLB\_LEGEND\_ON\_TOP  
LLB\_LEGEND\_POSITION  
LLB\_LEGEND\_VISIBLELOGICAL

## Axis

### AxisX

LLB_X_AXIS_COLOR	RGB value
LLB_X_AXIS_DATA_SOURCE	Pointer to a callback function
LLB_X_AXIS_HEADER	PSZ string
LLB_X_AXIS_HEADER_FONT	(Read Only) Font object
LLB_X_AXIS_LABEL_FONT	(Read Only) Font object
LLB_X_AXIS_ON_MOUSE	Pointer to a callback function
LLB_X_AXIS_TITLE	PSZ string
LLB_X_AXIS_TITLE_FONT	(Read Only) Font object

### AxisY Left and Right

LLB_Y_AXIS_AUTOSIZE	LOGICAL
LLB_Y_AXIS_AUTOSIZE_STEP	LOGICAL
LLB_Y_AXIS_BASE	DOUBLE
LLB_Y_AXIS_COLOR	RGB value
LLB_Y_AXIS_HEADER	PSZ string
LLB_Y_AXIS_HEADER_FONT	(Read Only) Font object
LLB_Y_AXIS_LABEL_FONT	(Read Only) Font object
LLB_Y_AXIS_MAX	DOUBLE
LLB_Y_AXIS_MIN	DOUBLE
LLB_Y_AXIS_ON_MOUSE	Pointer to a callback function
LLB_Y_AXIS_STEP_MAJOR	DOUBLE
LLB_Y_AXIS_STEP_MAJOR_TYPE	
LLB_Y_AXIS_STEP_MAJOR_COLOR	RGB value
LLB_Y_AXIS_STEP_MINOR	DOUBLE
LLB_Y_AXIS_STEP_MINOR_TYPE	
LLB_Y_AXIS_STEP_MINOR_COLOR	RGB value
LLB_Y_AXIS_TITLE	
LLB_Y_AXIS_TITLE_FONT	(Read Only) Font object
LLB_Y_AXIS_VALUE_TO_SCALE	Pointer to a callback function
LLB_Y_AXIS_VALUE_TO_STRING	Pointer to a callback function



## Column

LLB_COLUMN_ATTACHED_TO_Y_AXIS_LEFT	LOGICAL
LLB_COLUMN_COLOR	RGB value
LLB_COLUMN_COLORFILL	RGB value
LLB_COLUMN_COLORFILL_SHADOW	RGB value
LLB_COLUMN_DATA_SOURCE	Pointer to a callback function
LLB_COLUMN_FREEZE	LOGICAL
LLB_COLUMN_GET_POS_IN_GRAPH	(Read Only) SHORTINT
LLB_COLUMN_ON_MOUSE	Pointer to a callback function
LLB_COLUMN_PEN_WIDTH	SHORTINT from 1 - 5
LLB_COLUMN_TITLE_FONT	(Read Only) Font object
LLB_COLUMN_TITLE_LEFT	PSZ string
LLB_COLUMN_TITLE_RIGHT	PSZ string
LLB_COLUMN_X_WIDTH	SHORTINT from 0 - 100
LLB_COLUMN_X_OFFSET	SHORTINT from -50 - 50
LLB_COLUMN_Y_WIDTH	SHORTINT from 0 - 100
LLB_COLUMN_Y_OFFSET	SHORTINT from -50 - 50

## Fonts

LLB_FONT_BOLD	LOGICAL
LLB_FONT_COLOR	An RGB value
LLB_FONT_ITALIC	LOGICAL
LLB_FONT_NAME	A PSZ string
LLB_FONT_UNDERLINE	LOGICAL

## Graph

LLB_GRAPH_COORD_BOTTOM	SHORTINT
LLB_GRAPH_COORD_HEIGHT	SHORTINT
LLB_GRAPH_COORD_LEFT	SHORTINT
LLB_GRAPH_COORD_MAXIMIZE	LOGICAL
LLB_GRAPH_COORD_RIGHT	SHORTINT
LLB_GRAPH_COORD_TOP	SHORTINT
LLB_GRAPH_COORD_WIDTH	SHORTINT
LLB_GRAPH_DEVICE_CONTEXT	Device context
LLB_GRAPH_ERASE_TEXT_AXIS	LOGICAL
LLB_GRAPH_FIRST_COLUMN_IN_VIEW	Pointer to a column object
LLB_GRAPH_GO_TOP	Pointer to a callback function
LLB_GRAPH_GO_BOTTOM	Pointer to a callback function
LLB_GRAPH_INVALIDATE_DATA	(Read Only) LOGICAL
LLB_GRAPH_IS_STABLE	
LLB_GRAPH_LAST_ERROR	(Read Only) LLB_NO_ERROR
LLB_GRAPH_LEGEND	(Read Only) Legend object
LLB_GRAPH_MARGIN_LEFT	SHORTINT
LLB_GRAPH_MARGIN_TOP	SHORTINT
LLB_GRAPH_MARGIN_RIGHT	SHORTINT
LLB_GRAPH_MARGIN_BOTTOM	SHORTINT
LLB_GRAPH_NUM_2_COLUMN_IN_GRAPH	Pointer to a column object
LLB_GRAPH_NUM_2_COLUMN_IN_VIEW	Pointer to a column object
LLB_GRAPH_NUM_COL_IN_GRAPH	(Read Only) SHORTINT
LLB_GRAPH_NUM_COL_IN_VIEW	SHORTINT 1 - MaxCol
LLB_GRAPH_NUM_ROW_IN_VIEW	SHORTINT 1 - MaxRow
LLB_GRAPH_ON_ERROR	Pointer to a callback function
LLB_GRAPH_ON_MOUSE	Pointer to a callback function
LLB_GRAPH_SKIPPER	Pointer to a callback function
LLB_GRAPH_SKIPPER_ROTATE	
LLB_GRAPH_SIDE_XY_COLORFILL	RGB value
LLB_GRAPH_SIDE_XY_3D_EFFECT	LOGICAL
LLB_GRAPH_SIDE_XY_GRID	LOGICAL
LLB_GRAPH_SIDE_XY_TRANSPARENT	LOGICAL
LLB_GRAPH_SIDE_ZY_COLORFILL	RGB value
LLB_GRAPH_SIDE_ZY_3D_EFFECT	LOGICAL
LLB_GRAPH_SIDE_ZY_GRID	LOGICAL
LLB_GRAPH_SIDE_ZY_TRANSPARENT	LOGICAL
LLB_GRAPH_SIDE_ZX_COLORFILL	RGB value
LLB_GRAPH_SIDE_ZX_3D_EFFECT	LOGICAL
LLB_GRAPH_SIDE_ZX_GRID	LOGICAL
LLB_GRAPH_SIDE_ZX_TRANSPARENT	LOGICAL
LLB_GRAPH_STATE_BUILD	
LLB_GRAPH_STATE_DISPLAY	
LLB_GRAPH_STATE_INVALID	
LLB_GRAPH_STATE_REFRESH_DATA	
LLB_GRAPH_STATE_STABLE	
LLB_GRAPH_TITLE	PSZ string
LLB_GRAPH_TITLE_FONT	(Read Only) Font object
LLB_GRAPH_TYPE	
LLB_GRAPH_TYPE_PARAM1	LOGICAL
LLB_GRAPH_TYPE_PARAM2	LOGICAL
LLB_TYPE_BAR_GRAPH	3D/FILL
LLB_TYPE_LINE_GRAPH	3D/FILL
LLB_TYPE_STACKED_BAR_GRAPH	3D/% - Absolute

LLB_TYPE_PIE_GRAPH	
LLB_GRAPH_X_AXIS	(Read Only) AxisX object
LLB_GRAPH_Y_AXIS_LEFT	(Read Only) AxisYLeft object
LLB_GRAPH_Y_AXIS_RIGHT	(Read Only) AxisYRight object
LLB_GRAPH_Y_TO_X_RATIO	SHORTINT 1 - 200
LLB_GRAPH_Z_TO_X_ANGLE	SHORTINT 0-89
LLB_GRAPH_Z_TO_X_RATIO	SHORTINT 0 - 200
LLB_GRAPH_WINDOW_BK_COLOR	RGB value
LLB_GRAPH_WINDOW_HANDLE	Window handle

## Error

LLB\_ERROR  
LLB\_ERROR\_COLUMN\_ALLREADY\_FROZEN  
LLB\_ERROR\_COLUMN\_NOT\_FROZEN  
LLB\_ERROR\_COLUMN\_NOT\_IN\_GRAPH  
LLB\_ERROR\_COLUMN\_NOT\_IN\_VIEW  
LLB\_ERROR\_DEVICE\_UNDEFINED  
LLB\_ERROR\_FUNCTION\_GET\_DATA\_UNDEFINED  
LLB\_ERROR\_FUNCTION\_GO\_TOP\_UNDEFINED  
LLB\_ERROR\_FUNCTION\_GO\_BOTTOM\_UNDEFINED  
LLB\_ERROR\_FUNCTION\_SKIP\_UNDEFINED  
LLB\_ERROR\_INVALID\_DEFINE  
LLB\_ERROR\_INVALID\_PARAMETERS  
LLB\_ERROR\_INVALID\_PERCENTAGE  
LLB\_ERROR\_INVALID\_VALUE  
LLB\_ERROR\_INVALID\_DISPLAY\_AREA  
LLB\_ERROR\_INVALID\_WINDOW\_HANDLE  
LLB\_ERROR\_INVALID\_Y\_AXIS\_VALUE  
LLB\_ERROR\_INVALID\_Y\_AXIS\_STEP\_MAJOR  
LLB\_ERROR\_INVALID\_Y\_AXIS\_STEP\_MINOR  
LLB\_ERROR\_NO\_COLUMN\_IN\_GRAPH  
LLB\_ERROR\_NOT\_ENOUGH\_MEMORY  
LLB\_ERROR\_NULL\_OBJECT\_POINTER  
LLB\_ERROR\_UNKNOWN\_OBJECT

## **Introduction**

### **Welcome to Light Lib Business!**

Light Lib Business was designed to be the easiest-to-use business graphing library available for Windows. You can finally add powerful graphing capabilities to your applications, easily. Light Lib Business was developed with the following goals in mind:

#### **Ease of Use**

It is easy to integrate Light Lib Business into existing applications. You simply open a DBF, create a graph object, add some fields or columns from the DBF to the graph object and start browsing. That's it. Light Lib Business is set up with intelligent default values for the graph object, but of course, you can edit these values. This allows you to incrementally enhance business graphs without having to first learn many new programming techniques and terms.

#### **Execution speed**

Execution speed is excellent. The Light Lib Business data browsing kernel is written in assembly language. It implements smart graph stabilization for optimal performance and is able to keep the graph and the data synchronized efficiently. Light Lib Business is based on proven graphical DOS technology.

### **Using Light Lib Business**

Managing graphs and data is very complex. The reason Light Lib Business is so easy to implement in your applications is because it uses and takes full advantage Object Oriented Programming (OOP). Everything that makes up a graph, including the graph itself, is an "object" and each object is completely configurable.

The easiest way to learn any new concept is by example. Each installed language support has the source code to its own set of demo programs. Please reference them to gain a good understanding of how to use Light Lib Business.

### **Trademarks**

All trademarks are the property of their registered owners.

## **How to use this Help**

This help system was designed to provide quick access to information. Help is provided for the extensive language support and for the supplied Light Lib Business and Light Lib Objects DLLs.

**We strongly suggest that you use the individual language support with your applications!**

When a language is selected, you will be prompted with an overview of all support classes and/or functions. There is also a "How Do I?" section which provides step-by-step instructions on various common tasks.

A secondary window will open containing details and descriptions when any of these items are selected. This window is set to always stay on top. That way once a help topic is selected, you can continue working without losing focus on this window. To close it, simply select the window's system menu and select Close.

## Quick Start

Sample applications are provided for each supported language. You should execute the sample applications and experiment with the graph features in order to gain a good understanding of how Light Lib Business works. Once you understand how to maneuver through graph data, zoom in and out of a graph, change graph types, you will be able to easily modify the samples to fit your needs.

See also [How Do I?](#)



## Overview

Light Lib Business is a powerful yet easy to implement business graphing library for Windows. The basic concept is to provide end users the ability to dynamically navigate through a data source while displaying a graphical representation of the data values in the form of a graph.

Traditional graphing products force programmers, to tediously load data, calculate scaling values and window coordinates and adjust text, etc. before displaying even a simple graph. This process has to be repeated each time new values need to be displayed or a window is resized. Furthermore, the graph displayed by these products is merely an image or picture of the values. The end user is unable to interact with these graphs without substantial programming.

Light Lib Business introduces a much better and easier concept. Light Lib Business requires you to supply a data source in the form of an array or database and it will dynamically graph the data for you. Not only, is a graph displayed, it is also fully ready for end user interaction. Default dialogs are provided to allow end-users the ability to change any aspect of a graph simply by clicking over any part of a graph. This level of intelligent interaction between the data, end user and the graph itself, is possible because each graph is fully object oriented. Light Lib Business processes all mouse activity via a replaceable mouse handling system which is able to determine over which object in a graph the mouse action occurred.

A Light Lib Business graph is comprised of several major components or objects:

1. Graph object
2. Axis objects (3 - AxisX, AxisYLeft, AxisYRight)
3. One or more Column objects
4. Legend object

Each object is linked to the Graph object in an ownership relationship, (ie. an Axis, Column or Legend object can't exist without a Graph object). Some objects use sub-objects, such as Legend objects which use Font objects.

The callback functions (such as the mouse handling functions) receive several parameters which provide important programming flexibility, allowing you to customize actions (such as calling dialogs, displaying other windows, etc.).

## Native Language Support

**We strongly suggest that you use the native language support provided for each language, as opposed to calling the DLLs directly.**

We have provided these support layers (class hierarchies, VBXs, source code) as a simple way for you to use the product. If you need to go beyond the capabilities provided, you may need to call the DLLs directly, however most developers will never need to do so.

Wrapping is a term used to describe the process of encapsulating or hiding a systems complexities such as a DLL or Library. This may involve adding a layer of functions or classes to communicate between systems. A system could be an application, library, DLL or simply a group of functions or classes.

## Readme

## **Compatibility**

### **Windows Screen Drivers**

Light Lib Business is compatible with all installed Windows screen drivers.

### **Windows Printer Drivers**

Light Lib Business is compatible with all installed Windows printer drivers.

## Installed Files

The following is a list of installed files

C:\LLB	Installed directory of Light Lib Business
LLB.DLL	Light Lib Business DLL
LLBDEMO.EXE	Demonstration of Light Lib Images
LLB.HLP	Windows Help
LLB.ICO	Icon
LLB.WRI	Readme in Write format

C:\LLB\CAVO	CA-Visual Objects directory
LLBINTOP.AEF	Sample of Images in a Top Application Window
LLBVOSUP.AEF	CA-Visual Objects LLB System Classes

C:\LLB\MSC	Microsoft C/C++ directory
README.TXT	

C:\LLB\MSVB	Microsoft Visual Basic directory
README.TXT	
LLB.VBX	Sample of a Business graph using VBX

C:\LLB\DATA	Data used by all sample applications
-------------	--------------------------------------

## What's New...

## Appendices

No Help available for this section.



## CA-Visual Objects

The CA-Visual Objects support AEF supplied with Light Lib Business should not be modified directly since this support layer calls the Light Lib Business DLL directly.

### How Do I?

#### Classes

[AxisX](#)

[AxisY](#)

[AxisYLeft](#)

[AxisYRight](#)

[BaseFont](#)

[DBFDataSource](#)

[DBSDataSource](#)

[GraphColumn](#)

[GraphInWindow](#)

[GraphWindow](#)

[Legend](#)

#### Functions

[dwLightLibApp\(\)](#)

[dwLightLibAppRegister\(\)](#)

[dwLightLibAppUnRegister\(\)](#)

[IsGraph\(\)](#)

[IsGraphAxisLeftY\(\)](#)

[IsGraphAxisRightY\(\)](#)

[IsGraphAxisX\(\)](#)

[isGraphColumn\(\)](#)

[IsGraphLegend\(\)](#)

[StdMouseHandler\(\)](#)

#### Sample Graphs

[Simple Graph](#)

[Complex Graph](#)

## How Do I?

## CA-Visual Objects

### General

[Add a graph to my application](#)

[Add or remove columns in a graph](#)

[Change and mix graph types](#)

### System

[Register and unregister an application](#)

## How Do I?

## CA-Visual Objects

### Add a graph to my application

1. Create a Window object which is used for error handling
2. Register the application with the DLLs and pass the Window object
3. Create a data source which is a type of DBServer
4. Create a GraphWindow and pass the oDataSource
5. Show the GraphWindow
6. At the end of execution, unregister the application with the DLLs

### Sample Code

#### METHOD Start() CLASS App

```
LOCAL oWindow AS StandardShellWindow  
LOCAL oServer AS DBFDataSource  
LOCAL oGraph AS GraphWindow  
LOCAL sFile:="c:\sc_sp\samples\people" AS STRING  
  
// Create a ShellWindow  
oWindow := StandardShellWindow{ self }  
  
// Create a DBFDataSource  
oServer := DBFDataSource{ sFile }  
  
// Register this application with Light Lib Objects  
dwLightLibAppRegister( self, oWindow )  
  
// Create a GraphWindow  
oGraph := GraphWindow{ oWindow, oServer }  
  
// Call the autoLayout method. This makes assumptions  
// about the field layouts in the data. if these assumptions  
// do not match your data then you can add columns manually  
// using the addColumn method  
oGraph:autoLayout( oServer )  
  
// Display the ShellWindow  
oWindow:Show()  
  
// Display the GraphWindow  
oGraph:Show()  
  
// Execute the application  
self:Exec()  
  
// Unregister this application with Light Lib Objects  
dwLightLibAppUnregister( )
```

For another example, please see the [sample](#) code.

## How Do I?

## CA-Visual Objects

### Add or remove columns in a graph

1. Create a DataSource
2. Create a GraphWindow
3. Add the columns

### Sample Code

#### METHOD Start() CLASS App

```
LOCAL oWindow AS StandardShellWindow
LOCAL oServer AS DBFDataSource
LOCAL oGraph AS GraphWindow
LOCAL nl AS Word
LOCAL oColumn AS GraphColumn
LOCAL sFile := "j:\develop\data\sales.dbf"
```

```
// Create a ShellWindow
```

```
oWindow := StandardShellWindow{self}
```

```
// Create a DataSource
```

```
oServer := DBFDataSource{ sFile }
```

```
// Register this application with Light Lib Objects
```

```
dwLightLibAppRegister( self, oWindow )
```

```
// Create a GraphWindow
```

```
oGraph := GraphWindow{ oWindow, oServer }
```

```
// Loop through each field, creating a new column for each one
```

```
FOR nl := 2 TO oServer:FCOUNT
```

```
    oColumn := oGraph:graph:addColumn( {|siParam, oDataSource| ;
    oDataSource:fieldGet(siParam) }, nl, oServer )
```

```
// Assign a title to each column.
```

```
// This will be displayed in the graph Legend
```

```
oColumn:title := "Column " + LTRIM( ASSTRING( nl ) )
```

#### NEXT

```
// Display the ShellWindow
```

```
oWindow:Show()
```

```
// Display the GraphWindow
```

```
oGraph:Show()
```

```
// Run the application
```

```
self:Exec()
```

```
// Unregister this application from Light Lib Objects
```

```
dwLightLibAppUnRegister()
```

```
RETURN NIL
```

For further information, please see the [sample](#) code.

## How Do I?

## CA-Visual Objects

### Change and mix graph types

1. Create a DataSource
2. Create a GraphWindow
3. Add columns to the graph
4. Change the graph column type of selected columns

### Sample Code

#### METHOD Start() CLASS App

```
LOCAL oWindow AS StandardShellWindow  
LOCAL oServer AS DBFDataSource  
LOCAL oGraph AS GraphWindow  
LOCAL oColumn AS GraphColumn  
LOCAL sFile := "j:\develop\data\sales.dbf" AS STRING  
  
// Create a StandardShellWindow  
oWindow := StandardShellWindow{ self }  
  
// Create a DataSource  
oServer := DBFDataSource{ sFile }  
  
// Register this application with Light Lib Objects  
dwLightLibAppRegister( self, oWindow )  
  
// Create a GraphWindow  
oGraph := GraphWindow{ oWindow, oServer }  
  
// AutoLayout the graph, creating all columns, etc.  
oGraph:autoLayout()  
  
// Get a reference to the second column  
oColumn := oGraph:graph:getColumnInGraph(2)  
  
// Convert the column to a line type  
oColumn:TypeLine()  
  
// Don't fill it  
oColumn:EffectFilled := FALSE  
  
// Display the ShellWindow  
oWindow:Show()  
  
// Display the GraphWindow  
oGraph:Show()  
  
// Run the application  
self:Exec()  
  
// Unregister this application from Light Lib Objects  
dwLightLibAppUnregister()
```

**RETURN NIL**



**How Do I?**

**CA-Visual Objects**

[Change the default dialog windows](#)



## AxisX Class

### Purpose

Graph X Axis class

### Properties

[Color Access/Assign](#)

[DataSource Export](#)

[GetLabelArray Access/Assign](#)

[Header Access/Assign](#)

[HeaderFont Export](#)

[LabelFont Export](#)

[OnError Access/Assign](#)

[OnMouse Access/Assign](#)

[Title Access/Assign](#)

[TitleFont Export](#)

### Methods

[Destroy\(\)](#)

[Init\(\)](#)

### System Properties

*These properties are used internally. They are provided as reference only and should NEVER be accessed directly in your applications.*

[cbGetLabel Export](#)

[cbOnError Export](#)

[cbOnMouse Export](#)

[oGraph Export](#)

[siGetLabel Export](#)

### Inherits From

(No ancestors)

### Inherited By

(No descendants)

## AxisX:cbGetLabel Export

### Description

GetLabel codeblock. Do not assign a value directly, instead use [GetLabelArray Access/Assign](#)

## AxisX:cbOnError Export

### Description

OnError codeblock. Do not assign a value directly, instead use [OnError Access/Assign.](#)

## AxisX:cbOnMouse Export

### Description

OnMouse codeblock. Do not assign a value directly, instead use [OnMouse Access/Assign](#)

## **AxisX:Color Access/Assign**

### **Description**

Get/Set the X Axis color.

### **Type**

OBJECT Color

## **AxisX:Header Access/Assign**

### **Description**

Get/Set the X Axis header

### **Type**

STRING

## AxisX:GetLabelArray Access/Assign

### Description

Get/Set the GetLabel array which contains 3 elements. This is used to get a value from the data source and use it as labels for the rows or records. The array elements are:

- 1.Codeblock which receives the following two parameters at eval time
- 2.Optional SHORTINT to be passed to the codeblock. (ie column number for FieldGet() )
- 3.Optional data source object to be passed to the codeblock. If this value is not provided, the graph's data source will be used by default.

### Type

ARRAY

### Example

```
// This uses the first column in the data source as
// the Xlabel. In this case, siParam is the fieldnumber

cbLabel := { |siParam,oDataSource| ;
             AsString( oDataSource:FieldGet( siParam ) ) }

oAxisX:GetLabelArray := { cbLabel, 1, oDataSource }
```

## **AxisX:OnError Access/Assign**

### **Description**

Get/Set a codeblock which is evaluated when an error occurs on the X Axis.

### **Type**

CODEBLOCK



## **AxisX:OnMouse Access/Assign**

### **Description**

Get/Set a codeblock which is evaluated when a mouse event occurs on the X Axis. See [OnMouse Codeblock](#) .

### **Type**

CODEBLOCK

**AxisX:Title Access/Assign**

**Description**

Get/Set the title for X Axis.

**Type**

STRING

## AxisX:HeaderFont Export

### Description

Font used to display the X Axis header.

### Type

OBJECT [BaseFont](#)

## AxisX:LabelFont Export

### Description

Font used to display the X Axis label.

### Type

OBJECT [BaseFont](#)

## AxisX:DataSource Export

### Description

Data source object for the X Axis labels. This should be a type of data server. If a data source is not provided, the first field in the graph's [data server](#) will be used.

An example of using a different data source for the axis labels would be if you had an array of months.

### Type

OBJECT

## AxisX:oGraph Export

### Description

Reference to the Light Lib Business graph object which this AxisX object belongs to.

### Type

OBJECT [GraphInWindow](#)

## AxisX:siGetLabel Export

### Description

Value passed to the GetLabelArray. Do not assign a value directly, instead use [GetLabelArray Access/Assign](#).

### Type

SHORTINT

## **AxisX:TitleFont Export**

### **Description**

Font used to display the X Axis title.

### **Type**

OBJECT [BaseFont](#)



## **AxisX:Destroy() Method**

### **Purpose**

Destroy the X Axis object.

### **Syntax**

<oAxisX>:Destroy() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

Free all resources allocated to this object. This method is called automatically when the containing object is destroyed. You do not need to call this method directly.

## **AxisX:Init() Method**

### **Purpose**

Initialize the X Axis object.

### **Syntax**

AxisX{ <oGraph> } ---> SELF

### **Arguments**

<oGraph>	The graph object
----------	------------------

### **Returns**

SELF	A reference to a new AxisX object
------	-----------------------------------

### **Description**

There is no need to call this directly since the X Axis is automatically initialized when a graph is created.

## AxisY Class

### Purpose

Y Axis object

### Properties

[AutoSize Access/Assign](#)

[Base Access/Assign](#)

[Color Access/Assign](#)

[Header Access/Assign](#)

[HeaderFont Export](#)

[LabelFont Export](#)

[Max Access/Assign](#)

[Min Access/Assign](#)

[OnError Access/Assign](#)

[OnMouse Access/Assign](#)

[StepMajor Access/Assign](#)

[StepMajorColor Access/Assign](#)

[StepMajorType Access/Assign](#)

[StepMinor Access/Assign](#)

[StepMinorColor Access/Assign](#)

[StepMinorType Access/Assign](#)

[Title Access/Assign](#)

[TitleFont Export](#)

[ValueToScale Access/Assign](#)

[ValueToString Access/Assign](#)

### Methods

[Destroy\(\)](#)

[Init\(\)](#)

### System Properties

*These properties are used internally. They are provided as reference only and should NEVER be accessed directly in your applications.*

[cbOnError Export](#)

[cbOnMouse Export](#)

[cbValueToScale Export](#)

[cbValueToString Export](#)

[oGraph Export](#)

### Inherits From

(No ancestors)

### Inherited By

[AxisYLeft Class](#)

[AxisYRight Class](#)

## **AxisY:AutoSize Access/Assign**

### **Description**

Get/Set the AutoSizing feature. If set, the graph will automatically be resized to fit the greatest value in view from the data source. The Minor and Major steps are updated to reflect the values in view. This feature is extremely useful when the end-user is allowed to dynamically navigate through a data source.

### **Type**

LOGICAL

## **AxisY:Base Access/Assign**

### **Description**

Get/Set the Y Axis base value. This is the baseline value which falls between the minimum and maximum Y Axis values in a graph. The default is 0.

### **Type**

REAL8

## AxisY:cbOnError Export

### Description

OnError codeblock. Do not assign a value directly, instead use [OnError Access/Assign.](#)

## AxisY:cbOnMouse Export

### Description

OnMouse codeblock. Do not assign a value directly, instead use [OnMouse Access/Assign](#)

## AxisY:cbValueToScale Export

### Description

ValueToScale codeblock. Do not assign a value directly, instead use [ValueToScale Access/Assign.](#)



## AxisY:cbValueToString Export

### Description

ValueToString codeblock. Do not assign a value directly, instead use [ValueToString Access/Assign](#).

## **AxisY:Color Access/Assign**

### **Description**

Get/Set the Y Axis color. The Major and Minor steps are affected.

### **Type**

OBJECT Color

## **AxisY:Header Access/Assign**

### **Description**

Get/Set the Y Axis header.

### **Type**

STRING

## AxisY:HeaderFont Export

### Description

Font used to display the Y Axis header.

### Type

OBJECT [BaseFont](#)

## AxisY:LabelFont Export

### Description

Font used to display the Y Axis label.

### Type

OBJECT [BaseFont](#)

**AxisY:Max Access/Assign****Description**

Get/Set the Y Axis maximum scale value.

**Type**

REAL8

**AxisY:Min Access/Assign****Description**

Get/Set the Y Axis minimum scale value.

**Type**

REAL8

## AxisY:oGraph Export

### Description

Reference to the Light Lib Business graph object which this AxisY object belongs to.

### Type

OBJECT [GraphInWindow](#)



## **AxisY:OnError Access/Assign**

### **Description**

Get/Set a codeblock which is evaluated when an error occurs on the Y Axis.

### **Type**

CODEBLOCK

## **AxisY:OnMouse Access/Assign**

### **Description**

Get/Set a codeblock which is evaluated when a mouse event occurs on the Y Axis. See [OnMouse Codeblock](#) .

### **Type**

CODEBLOCK

**AxisY:StepMajor Access/Assign****Description**

Get/Set the YAxis major step value.

The Y Axis has two types of steps which are used as visual markers for the column values. Major steps represent the larger increments. Minor steps represent the increments between the Major steps.

**Type**

REAL8

## **AxisY:StepMajorColor Access/Assign**

### **Description**

Get/Set the color used to display the major steps on the Y Axis.

### **Type**

OBJECT Color

## **AxisY:StepMajorType Access/Assign**

### **Description**

Get/Set the Y Axis major step type. There are no optional types available at this time.

## **AxisY:StepMinor Access/Assign**

### **Description**

Get/Set the YAxis minor step value

The Y Axis has two types of steps which are used as visual markers for the column values. Major steps represent the larger increments. Minor steps represent the increments between the Major steps.

### **Type**

REAL8

## **AxisY:StepMinorColor Access/Assign**

### **Description**

Get/Set the color used to display the minor steps on the Y Axis.

### **Type**

OBJECT Color

## **AxisY:StepMinorType Access/Assign**

### **Description**

Get/Set the Y Axis minor type. There are no optional types available at this time.



**AxisY:Title Access/Assign**

**Description**

Get/Set the Y Axis title.

**Type**

STRING

**AxisY:TitleFont Access/Assign**

**Description**

Font used to display the Y Axis title.

**Type**

OBJECT [BaseFont](#)

## **AxisY:ValueToScale Access/Assign**

### **Description**

Codeblock used to scale Y Axis label values. This is useful in providing a mechanism to scale values in the graph. For example, divide all graph values by 1000.

### **Type**

CODEBLOCK

### **Example**

```
oAxisY:ValueToScale := { | r8Value | r8Value / 1000 }
```

## **AxisY:ValueToString Access/Assign**

### **Description**

Codeblock used to transform the Y Axis label values into strings for display purposes.

### **Type**

CODEBLOCK

### **Example**

```
oAxisY:ValueToString := { | nValue | TRANSFORM( nValue , "99,999.99" ) }
```

## **AxisY:Destroy() Method**

### **Purpose**

Destroy the AxisY object.

### **Syntax**

<oAxisY>:Destroy() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

Free all resources allocated to this object. This method is called automatically when the containing object is destroyed. You do not need to call this method directly.

## AxisY:Init() Method

### Purpose

Initialize a Y Axis object.

AxisY is an abstract class and should not be used directly; instead, use one of its subclasses, which include [AxisYLeft](#) and [AxisYRight](#) .

### Syntax

```
AxisY{ <oGraph>, <lLeftAxis> } ---> SELF
```

### Arguments

<oGraph>	Reference to the graph object which contains this Axis.
<lLeftAxis>	If the Left or Right Y Axis is to be initialized.

### Returns

SELF	A reference to a new AxisYobject.
------	-----------------------------------

## AxisYLeft Class

### Purpose

Left AxisY object.

### Properties

None

### Methods

[Init\(\)](#)

### Inherits From

[AxisY Class](#)

### Inherited By

(No descendants)

## **AxisYLeft:Init() Method**

### **Purpose**

Initialize the left AxisY object. There is no need to call this directly since the left Y Axis is automatically initialized when a graph is created.

### **Syntax**

```
AxisYLeft{ <oGraph> } ---> SELF
```

### **Arguments**

<i>&lt;oGraph&gt;</i>	Reference to the graph object which contains this Axis.
-----------------------	---

### **Returns**

SELF	A reference to a new AxisYLeft object.
------	--

### **Description**

Light Lib Business allows you to attach columns to either the left or right Y Axis. By default, all columns are attached to the left Y Axis. Having two Y axis is useful, for example, when a different scale is needed to display columns.



## AxisYRight Class

### Purpose

Right AxisY object.

### Properties

None

### Methods

[Init\(\)](#)

### Inherits From

[AxisY Class](#)

### Inherited By

(No descendants)

## **AxisYRight:Init() Method**

### **Purpose**

Initialize the right AxisY object. There is no need to call this directly since the right Y Axis is automatically initialized when a graph is created.

### **Syntax**

```
AxisYRight{ <oGraph> } ---> SELF
```

### **Arguments**

<i>&lt;oGraph&gt;</i>	Reference to the graph object which contains this Axis.
-----------------------	---

### **Returns**

SELF	A reference to a new AxisYRight object.
------	---

### **Description**

Light Lib Business allows you to attach columns to either the left or right Y Axis. By default, all columns are attached to the left Y Axis. Having two Y axis is useful, for example, when a different scale is needed to display columns.

## BaseFont Class

### Purpose

The BaseFont Class determines text attributes.

### Properties

[Bold Access/Assign](#)

[Color Access/Assign](#)

[Italic Access/Assign](#)

[Name Access/Assign](#)

[Underline Access/Assign](#)

### Methods

[Init\(\)](#)

### Inherits From

(No ancestors)

### Inherited By

(No descendants)

**BaseFont:Bold Access/Assign**

**Description**

Logical flag which determines whether this font is in bold or not.

**Type**

LOGICAL

## **BaseFont:Color Access/Assign**

### **Description**

Get/Set the font's color attribute.

### **Type**

OBJECT Color

**BaseFont:Italic Access/Assign**

**Description**

Logical flag which determines whether this font is in italic or not.

**Type**

LOGICAL

**BaseFont:Name Access/Assign**

**Description**

Get/Set the font name.

**Type**

STRING

**BaseFont:Underline Access/Assign****Description**

Logical flag which determines whether this font is underlined or not.

**Type**

LOGICAL



## BaseFont:Init() Method

### Purpose

Initialize the font object.

### Syntax

```
BaseFont{ <dwLLBusinessOwner>, <liFontObject> } ---> SELF
```

### Arguments

<dwLLBusinessOwner> Reference to the graph object

<liFontObject>               Reference to the graph sub-object

### Returns

SELF                           A reference to a new BaseFont object

### Description

The BaseFont class is not meant to be instantiated directly by the application, and you would never have a BaseFont object which does not have a counterpart object at the DLL level.

BaseFont objects are created automatically when you create a graph. For instance, when you create a graph, an X and Y axis and legend is automatically created for that graph. When this axis is created it in turn will create three BaseFont objects associated with the text parts of the axis (label, header and title). liFontObject is an identifier for the sub-object (ie. Title).

All font (BaseFont) objects needed by a graph are created once the graph is created. You would access the font object through the variable in the client class (ie. AxisX:titleFont).

## DBFDataSource Class

### Purpose

Basic sample of a custom data server which does not inherit from DataServer{ }.

### Properties

[FCount Access](#)

[RecCount Access](#)

[RecNo Access](#)

### Methods

[Close\(\)](#)

[FieldGet\(\)](#)

[FieldName\(\)](#)

[GoBottom\(\)](#)

[GoTo\(\)](#)

[GoTop\(\)](#)

[Init\(\)](#)

[Select\(\)](#)

[Skipper\(\)](#)

[UnSelect\(\)](#)

### Inherits From

(No ancestors)

### Inherited By

(No descendants)

### Notes

In order to use Light Lib Business, an object oriented datasource such as DataServer{ } is needed. If a custom DataServer{ } is used, then the following methods **must** be provided:

Skipper()	Returns the number of rows or records skipped.
GoTop()	Return True if successfull
GoBottom()	Return True if successfull

## **DBFDataSource:FCount Access**

### **Description**

Number of columns or fields in the data source

### **Type**

INTEGER

## **DBFDataSource:RecCount Access**

### **Description**

Number of rows or records in the data source

### **Type**

INTEGER

**DBFDataSource:RecNo Access**

**Description**

Current row or record number position.

**Type**

INTEGER

## **DBFDataSource:Close() Method**

### **Purpose**

Close the data source

### **Syntax**

<oDBFDataSource>:Close() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **DBFDataSource:FieldGet() Method**

### **Purpose**

Return the field data of a specified column or field number

### **Syntax**

`<oDBFDataSource>.FieldGet( <nCol> ) ---> <xValue>`

### **Arguments**

<code>&lt;nCol&gt;</code>	Column or field number in the data source
---------------------------	---

### **Returns**

<code>&lt;xValue&gt;</code>	Field data
-----------------------------	------------

## **DBFDataSource:FieldName() Method**

### **Purpose**

Return the field name of a specified column or field number

### **Syntax**

```
<oDBFDataSource>:FieldName( <nCol> ) ---> <xValue>
```

### **Arguments**

<nCol/>	Column or field number in the data source
---------	---

### **Returns**

<xValue>	Field name
----------	------------



## **DBFDataSource:GoBottom() Method**

### **Purpose**

Move the record pointer to the bottom of the data source .

### **Syntax**

```
<oDBFDataSource>:GoBottom() ---> <IResult>
```

### **Arguments**

None

### **Returns**

<IResult>                      If the operation was successful

## **DBFDataSource:GoTo() Method**

### **Purpose**

Move the record pointer to a specified row or record number.

### **Syntax**

```
<oDBFDataSource>:GoTo( <nRow> ) ---> </Result>
```

### **Arguments**

<nRow>	Row or record number to move the record pointer to
--------	--

### **Returns**

</Result>	If the operation was successful
-----------	---------------------------------

## **DBFDataSource:GoTop() Method**

### **Purpose**

Move the record pointer to the top of the data source .

### **Syntax**

```
<oDBFDataSource>:GoTop() ---> <IResult>
```

### **Arguments**

None

### **Returns**

```
<IResult>
```

## **DBFDataSource:Init() Method**

### **Purpose**

Create a DBFDataSource object

### **Syntax**

```
DBFDataSource{ <sFileName> } ---> SELF
```

### **Arguments**

<sFileName>            DBF file name.

### **Returns**

SELF

## **DBFDataSource:Select() Method**

### **Purpose**

Select a work area.

### **Syntax**

<oDBFDataSource>:Select() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **DBFDataSource:Skipper() Method**

### **Purpose**

Skip a specified number of rows or records in the data source

### **Syntax**

<oDBFDataSource>:Skipper( *<nRowToSkip>* ) ---> NIL

### **Arguments**

*<nRowToSkip>*                Number of rows to skip. Default is 1.

### **Returns**

NIL

## **DBFDataSource:UnSelect() Method**

### **Purpose**

Select the last area.

### **Syntax**

```
<oDBFDataSource>.UnSelect() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

## DBSDataSource Class

### Purpose

Data server which inherits from DBServer{}

### Properties

None

### Methods

[Init\(\)](#)

[Skipper\(\)](#)

### Inherits From

(No ancestors)

### Inherited By

(No descendants)

### Notes

In order to use Light Lib Business, a DataServer containing a Skipper() Method that returns the number of rows skipped is required.



## **DBSDDataSource.Init() Method**

### **Purpose**

Create a DBSDDataSource object

### **Syntax**

```
DBFSDDataSource{ <cFileName> } ---> SELF
```

### **Arguments**

<cFileName>	Data server's file name
-------------	-------------------------

### **Returns**

SELF

## **DBSDDataSource:Skipper() Method**

### **Purpose**

Skip a specified number of rows or records in the data source

### **Syntax**

```
<oDBSDDataSource>:Skipper( <nRow> ) ---> <nSkipped>
```

### **Arguments**

<nRow>	Number of rows or records to skip
--------	-----------------------------------

### **Returns**

<nSkipped>	Number of rows or records actually skipped
------------	--

### **Notes**

In order to use Light Lib Business, a DataServer containing a Skipper() Method that returns the number of rows skipped is required.

## GraphColumn Class

### Purpose

Represent the values in the graph's data source.

### Properties

[AttachedToLeftAxis Access/Assign](#)

[AutoShadow Access/Assign](#)

[Color Access/Assign](#)

[ColorFill Access/Assign](#)

[ColorFill1 Access/Assign](#)

[ColorFill2 Access/Assign](#)

[ColorFill3 Access/Assign](#)

[ColorFill4 Access/Assign](#)

[ColorFillShadow Access/Assign](#)

[DataSource Export](#)

[Effect3D Access/Assign](#)

[EffectFilled Access/Assign](#)

[Freeze Access/Assign](#)

[GetValueArray Access/Assign](#)

[GetValueArray1 Access/Assign](#)

[GetValueArray2 Access/Assign](#)

[GetValueArray3 Access/Assign](#)

[GetValueArray4 Access/Assign](#)

[Name Export](#)

[OffsetX Access/Assign](#)

[OffsetY Access/Assign](#)

[OnError Access/Assign](#)

[OnMouse Access/Assign](#)

[PenWidth Access/Assign](#)

[PosInGraph Access/Assign](#)

[Title Access/Assign](#)

[TitleFont Export](#)

[Value Access](#)

[WidthX Access/Assign](#)

[WidthY Access/Assign](#)

### Methods

[Destroy\(\)](#)

[GetValueArrayGeneric\(\)](#)

[Init\(\)](#)

[IsColumnTypeBar\(\)](#)

[IsColumnTypeBarStock\(\)](#)

[IsColumnTypeLine\(\)](#)

[IsColumnTypePie\(\)](#)

[IsColumnTypeStacked\(\)](#)

[IsColumnTypeStackedPercent\(\)](#)

[MoveBackGraph\(\)](#)

[MoveBackView\(\)](#)

[MoveFrontGraph\(\)](#)

[MoveFrontView\(\)](#)

[MoveRelative\(\)](#)

[TypeBar\(\)](#)

[TypeBarStock\(\)](#)

[TypeLine\(\)](#)  
[TypePie\(\)](#)  
[TypeStacked\(\)](#)  
[TypeStackedPercent\(\)](#)

### **System Properties**

*These properties are used internally. They are provided as reference only and should NEVER be accessed directly in your applications.*

[cbGetValue1 Export](#)  
[cbGetValue2 Export](#)  
[cbGetValue3 Export](#)  
[cbGetValue4 Export](#)  
[cbOnError Export](#)  
[cbOnMouse Export](#)  
[oGraph Export](#)  
[siGetValue1 Export](#)  
[siGetValue2 Export](#)  
[siGetValue3 Export](#)  
[siGetValue4 Export](#)

### **Inherits From**

(No ancestors)

### **Inherited By**

(No descendants)

## **GraphColumn:AttachedToLeftAxis Access/Assign**

### **Description**

Logical flag which determines if the column is attached to left Y axis.

### **Type**

LOGICAL

## **GraphColumn:AutoShadow Access/Assign**

### **Description**

Logical flag which determines if the column is be shadowed.

### **Type**

LOGICAL

## **GraphColumn:Color Access/Assign**

### **Description**

Get/Set the column outline color.

### **Type**

OBJECT Color

## **GraphColumn:ColorFill Access/Assign**

### **Description**

Get/Set the color used to fill the column.

### **Type**

OBJECT Color



## **GraphColumn:ColorFill1 Access/Assign**

### **Description**

Get/Set the color used to fill the column's sub-value 1.

### **Type**

OBJECT Color

## **GraphColumn:ColorFill2 Access/Assign**

### **Description**

Get/Set the color used to fill the column's sub-value 2.

### **Type**

OBJECT Color

## **GraphColumn:ColorFill3 Access/Assign**

### **Description**

Get/Set the color used to fill the column's sub-value 3.

### **Type**

OBJECT Color

## **GraphColumn:ColorFill4 Access/Assign**

### **Description**

Get/Set the color used to fill the column's sub-value 4.

### **Type**

OBJECT Color

## **GraphColumn:ColorFillShadow Access/Assign**

### **Description**

Get/Set the color used to display the column shadow. If [GraphColumn:AutoShadow](#) is TRUE, this color will be automatically calculated.

### **Type**

OBJECT Color

## **GraphColumn:Effect3D Access/Assign**

### **Description**

Logical flag which determines if the column has a 3D appearance.

### **Type**

LOGICAL

## **GraphColumn:EffectFilled Access/Assign**

### **Description**

Logical flag which determines if the column should be filled in.

### **Type**

LOGICAL

## **GraphColumn:Freeze Access/Assign**

### **Description**

Logical flag which determines if the column is frozen. If a column is frozen, it will remain in view when panning across a graph's columns. Multiple columns may be frozen.

### **Type**

LOGICAL



## GraphColumn:GetValueArray Access/Assign

### Description

Get/Set the GetValueArray. This contains 3 elements and is used to get the data from the data source and use them as the row's values. GetValueArray's three elements are:

1. Codeblock which receives the following two parameters at eval time
2. Optional SHORTINT to be passed to the codeblock. ie column number for FieldGet()
3. Optional datasource to be passed to the codeblock. If this value is not provided, the graph's data source will be used by default.

### Type

ARRAY

### Example

```
// Add all the columns from the data source to the graph
// In this case, siParam is the field number

LOCAL cbGetValue, nI

For nI := 1 To oDataSource:FCount

    cbGetValue := { |siParam,oDataSource| ;
                  oDataSource:FieldGet( siParam ) }

    oGraph:AddColumn( cbGetValue, nI, oDataSource )

Next nI
```

## **GraphColumn:GetValueArray1 Access/Assign**

### **Description**

Get/Set the GetValueArray1. This contains 3 elements and is used to get the data from the data source and use them as the row's values. GetValueArray's three elements are:

1. Codeblock which receives the following two parameters at eval time
2. Optional SHORTINT to be passed to the codeblock. ie column number for FieldGet()
3. Optional datasource to be passed to the codeblock. If this value is not provided, the graph's data source will be used by default.

### **Type**

ARRAY

See [GraphColumn:GetValueArray](#)

## **GraphColumn:GetValueArray2 Access/Assign**

### **Description**

Get/Set the GetValueArray2. This contains 3 elements and is used to get the data from the data source and use them as the row's values. GetValueArray's three elements are:

1. Codeblock which receives the following two parameters at eval time
2. Optional SHORTINT to be passed to the codeblock. ie column number for FieldGet()
3. Optional datasource to be passed to the codeblock. If this value is not provided, the graph's data source will be used by default.

### **Type**

ARRAY

See [GraphColumn:GetValueArray](#)

## **GraphColumn:GetValueArray3 Access/Assign**

### **Description**

Get/Set the GetValueArray3. This contains 3 elements and is used to get the data from the data source and use them as the row's values. GetValueArray's three elements are:

1. Codeblock which receives the following two parameters at eval time
2. Optional SHORTINT to be passed to the codeblock. ie column number for FieldGet()
3. Optional datasource to be passed to the codeblock. If this value is not provided, the graph's data source will be used by default.

### **Type**

ARRAY

See [GraphColumn:GetValueArray](#)

## **GraphColumn:GetValueArray4 Access/Assign**

### **Description**

Get/Set the GetValueArray4. This contains 3 elements and is used to get the data from the data source and use them as the row's values. GetValueArray's three elements are:

1. Codeblock which receives the following two parameters at eval time
2. Optional SHORTINT to be passed to the codeblock. ie column number for FieldGet()
3. Optional datasource to be passed to the codeblock. If this value is not provided, the graph's data source will be used by default.

### **Type**

ARRAY

See [GraphColumn:GetValueArray](#)

## GraphColumn:cbGetValue1 Export

### Description

Reference to the GetValue1 codeblock. Do not assign directly, instead use [GraphColumn:GetValue1](#).

### Type

CODEBLOCK

## GraphColumn:cbGetValue2 Export

### Description

Reference to the GetValue2 codeblock. Do not assign directly, instead use [GraphColumn:GetValue2](#).

### Type

CODEBLOCK

## GraphColumn:cbGetValue3 Export

### Description

Reference to the GetValue3 codeblock. Do not assign directly, instead use [GraphColumn:GetValue3](#).

### Type

CODEBLOCK



## GraphColumn:cbGetValue4 Export

### Description

Reference to the GetValue4 codeblock. Do not assign directly, instead use [GraphColumn:GetValue4](#).

### Type

CODEBLOCK

## GraphColumn:cbOnMouse Export

### Description

OnMouse codeblock. Do not assign a value directly, instead use [GraphColumn:OnMouse](#)

## GraphColumn:cbOnError Export

### Description

OnError codeblock. Do not assign a value directly, instead use [GraphColumn:OnError](#)

## GraphColumn:Name Export

### Description

This is an optional column identifier of any data type, which is useful for providing a meaningful name to a column data source.

### Type

Any data type.

### Example

```
oGraph:getColumn(1):name := "JAPAN SALES"  
oGraph:getColumn(2):name := #US_SALES  
oGraph:getColumn(3):name := 1234
```

## **GraphColumn:oDataSource Export**

### **Description**

Data source for the column. Should be a type of data server and it can be independent of the graph's data source. If a column data source is not provided, the column will use the graph's data source.

### **Type**

OBJECT

## **GraphColumn:OffsetX Access/Assign**

### **Description**

Get/Set the column's X offset. Default is 0.

When the offset is 0, the middle of the column is centered in each row/column grid. The offset can range from -50 to +50. This allows a column to be shifted or offset within its displayable grid.

### **Type**

SHORTINT

## **GraphColumn:OffsetY Access/Assign**

### **Description**

Get/Set the column's Y offset. Default is 0.

When the offset is 0, the middle of the column is centered in each row/column grid. The offset can range from -50 to +50. This allows a column to be shifted or offset within its displayable grid

### **Type**

SHORTINT

## GraphColumn:oGraph Export

### Description

Reference to the graph object containing this column object.

### Type

OBJECT [GraphInWindow](#)



## **GraphColumn:OnError Access/Assign**

### **Description**

Get/Set a codeblock which is evaluated when an error occurs on the column.

### **Type**

CODEBLOCK

## **GraphColumn:OnMouse Access/Assign**

### **Description**

Get/Set a codeblock which is evaluated when a mouse event occurs on the column. See [OnMouse Codeblocks](#)

### **Type**

CODEBLOCK

## **GraphColumn:PenWidth Access/Assign**

### **Description**

Get/Set the column's pen width which is used to draw the outline for the column. The pen width ranges from 1 to 5 and the default is 1.

### **Type**

SHORTINT

## **GraphColumn:PosInGraph Access/Assign**

### **Description**

Get/Set the absolute position of the column in the graph. This value can be modified even when the column is not in view.

### **Type**

SHORTINT

## GraphColumn:siGetValue1 Export

### Description

Reference to the GetValue1 optional SHORTINT parameter. codeblock. Do not assign directly, instead use [GraphColumn:GetValue1](#).

### Type

SHORTINT

## GraphColumn:siGetValue2 Export

### Description

Reference to the GetValue1 optional SHORTINT parameter. codeblock. Do not assign directly, instead use [GraphColumn:GetValue2](#).

### Type

SHORTINT

## GraphColumn:siGetValue3 Export

### Description

Reference to the GetValue1 optional SHORTINT parameter. codeblock. Do not assign directly, instead use [GraphColumn:GetValue3](#).

### Type

SHORTINT

## GraphColumn:siGetValue4 Export

### Description

Reference to the GetValue1 optional SHORTINT parameter. codeblock. Do not assign directly, instead use [GraphColumn:GetValue4](#).

### Type

SHORTINT



## GraphColumn:TitleFont Export

### Description

Font used to display the column's title.

### Type

OBJECT [BaseFont](#)

**GraphColumn:Title Access/Assign**

**Description**

Get/Set the column's title.

**Type**

STRING

## **GraphColumn:Value Access**

### **Description**

Returns the column's current value from the data source.

## **GraphColumn:WidthX Access/Assign**

### **Description**

Get the column's X width. Default is 100.

The width ranges from 0 to 100 and the default is 100. This allows a column to be sized within its displayable grid. See [GraphColumn:OffsetX](#) and [GraphColumn:OffsetY](#)

### **Type**

SHORTINT

## **GraphColumn:WidthY Access/Assign**

### **Description**

Get the column's Y width. Default is 100.

The width ranges from 0 to 100 and the default is 100. This allows a column to be sized within its displayable grid. See [GraphColumn:OffsetX](#) and [GraphColumn:OffsetY](#)

### **Type**

SHORTINT

## **GraphColumn:TypeBar() Method**

### **Purpose**

Set the column to be a bar graph.

### **Syntax**

```
<oGraphColumn>.TypeBar() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Light Lib Business allows multiple graph types to be used by different columns in the same graph. For example, you can mix bar and line graph types in the same graph. Furthermore, each column maintains unique visual attributes such as the 3D and filled effects.

## **GraphColumn:TypeBarStock() Method**

### **Purpose**

Set the column to be a bar graph representing High and Low values (Stock Bar graph).

### **Syntax**

```
<oGraphColumn>.TypeBarStock() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Light Lib Business allows multiple graph types to be used by different columns in the same graph. For example, you can mix bar and line graph types in the same graph. Furthermore, each column maintains unique visual attributes such as the 3D and filled effects.

## **GraphColumn:TypeLine() Method**

### **Purpose**

Set the column to be a line graph.

### **Syntax**

```
<oGraphColumn>.TypeLine() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Light Lib Business allows multiple graph types to be used by different columns in the same graph. For example, you can mix bar and line graph types in the same graph. Furthermore, each column maintains unique visual attributes such as the 3D and filled effects.



## **GraphColumn:TypePie() Method**

### **Purpose**

Set the column to be a pie graph.

### **Syntax**

```
<oGraphColumn>.TypePie() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Each pie represents a column and each slice represents a value from the data source.

This type of graph does not allow mixing column graph types.

## **GraphColumn:TypeStacked() Method**

### **Purpose**

Set the column to be a stacked bar graph.

### **Syntax**

```
<oGraphColumn>.TypeStacked() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Each bar represents a column and each portion of the column represents a value from the data source. The total of all the values in a column is represented by the height of the bar.

This type of graph does not allow mixing column graph types.

## **GraphColumn:TypeStackedPercent() Method**

### **Purpose**

Set the column to be a stacked percentage bar graph with each column the same height.

### **Syntax**

```
<oGraphColumn>.TypeStackedPercent() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Each bar represents a column and each portion of the column represents a percentage of the total of all values.

This type of graph does not allow mixing column graph types.

## **GraphColumn:Destroy() Method**

### **Purpose**

Destroy the column object.

### **Syntax**

```
<oGraphColumn>:Destroy() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Free all resources allocated to this object. This method is called automatically when the containing object is destroyed. You do not need to call this method directly.

## GraphColumn:GetValueArrayGeneric() Method

### Purpose

Set/Clear the GetValueArray

### Syntax

<oGraphColumn>:GetValueArrayGeneric( <aGetValue>, <siSubColumn> ) ---> NIL

### Arguments

<aGetValue>                      Contains 3 elements which are used to get data from the datasource and use it as the rows values. The 3 elements are:

1. Codeblock which will receive the second and third elements as parameters at eval time
2. Optional ShortInt to be passed to the codeblock. ( ie. column number for FieldGet( # ) )
3. Optional datasource to be passed to the codeblock. If this value is not provided, the graph [GraphInWindow:oDataSource](#) will be used by default.

<siSubColumn> Sub-column to apply the aGetValue

### Returns

NIL

## GraphColumn:Init() Method

### Purpose

Initialize the column object.

### Syntax

```
GraphColumn{ <oGraph>, <cbGetValue>, <siGetValue>, <oDataSource> } ---> SELF
```

### Arguments

<oGraph>	The graph object
<cbGetValue>	See GetValueArray
<siGetValue>	Access/Assign for the column
<oDataSource>	Optional parameters

### Returns

SELF	Column object
------	---------------

## GraphColumn:IsColumnTypeBar() Method

### Purpose

Returns if the column is a bar graph.

### Syntax

```
<oGraphColumn>.IsColumnTypeBar() ---> </Value>
```

### Arguments

None

### Returns

</Value>	If the column is a bar graph
----------	------------------------------

## GraphColumn:IsColumnTypeBarStock() Method

### Purpose

Returns if the column is a bar stock graph.

### Syntax

```
<oGraphColumn>.IsColumnTypeBarStock() ---> </Value>
```

### Arguments

None

### Returns

</Value>	If the column is a bar stock graph
----------	------------------------------------



## GraphColumn:IsColumnTypeLine() Method

### Purpose

Returns if the column is a line graph.

### Syntax

```
<oGraphColumn>.IsColumnTypeLine() ---> </Value>
```

### Arguments

None

### Returns

</Value>	If the column is a line graph
----------	-------------------------------

## GraphColumn:IsColumnTypePie() Method

### Purpose

Returns if the column is a pie graph.

### Syntax

```
<oGraphColumn>.IsColumnTypePie() ---> </Value>
```

### Arguments

None

### Returns

</Value>	If the column is a pie graph
----------	------------------------------

## GraphColumn.IsColumnTypeStacked() Method

## Purpose

Returns if the column is a stacked bar graph.

## Syntax

```
<oGraphColumn>:IsColumnTypeStacked() ---> </Value>
```

## Arguments

None

## Returns

<code>&lt;/Value&gt;</code>	If the column is a stacked bar graph
-----------------------------	--------------------------------------

## GraphColumn:IsColumnTypeStackedPercent() Method

### Purpose

Returns if the column is a stacked percent bar graph.

### Syntax

```
<oGraphColumn>.IsColumnTypeStackedPercent() ---> </Value>
```

### Arguments

None

### Returns

</Value>                      If the column is a stacked percent bar graph

## **GraphColumn:MoveBackGraph() Method**

### **Purpose**

Move the column to the last position in the graph.

### **Syntax**

<oGraphColumn>:MoveBackGraph() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

Depending on the columns in view, the column may not be visible after being moved.

## **GraphColumn:MoveBackView() Method**

### **Purpose**

Move the column to the last position in the view.

### **Syntax**

```
<oGraphColumn>:MoveBackView() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

## **GraphColumn:MoveFrontGraph() Method**

### **Purpose**

Move the column to the first position in the graph.

### **Syntax**

<oGraphColumn>:MoveFrontGraph() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

Depending on the columns in view, the column may not be visible after being moved.

## **GraphColumn:MoveFrontView() Method**

### **Purpose**

Move the column to the first position in the view.

### **Syntax**

<oGraphColumn>:MoveFrontView() ---> NIL

### **Arguments**

None

### **Returns**

NIL



## **GraphColumn:MoveRelative() Method**

### **Purpose**

Move the column siMove positions in the graph.

### **Syntax**

<oGraphColumn>:MoveRelative( <siMove> ) ---> NIL

### **Arguments**

<siMove>	Number of columns to move the column object
----------	---

### **Returns**

NIL

### **Description**

Positive numbers move the column back and negative numbers move the column forward.

Depending on the columns in view, the column may not be visible after being moved.

## GraphWindow Class

### Purpose

Provide a window capable of displaying a [GraphInWindow object](#).

### Properties

[DataSource](#) [Export](#)

[Graph](#) [Export](#)

### Methods

[AutoLayout\(\)](#)

[Close\(\)](#)

[CloseGraph\(\)](#)

[DecAngleZToX\(\)](#)

[DecYToX\(\)](#)

[DecZToX\(\)](#)

[DefaultAspect\(\)](#)

[Destroy\(\)](#)

[Effect3D\(\)](#)

[EffectFilled\(\)](#)

[Expose\(\)](#)

[FileExit\(\)](#)

[HorizontalScroll\(\)](#)

[HorizontalScrollRefresh\(\)](#)

[IncAngleZToX\(\)](#)

[IncYToX\(\)](#)

[IncZToX\(\)](#)

[Init\(\)](#)

[MouseButtonDown\(\)](#)

[MoveDn\(\)](#)

[MoveGoBottom\(\)](#)

[MoveGoTop\(\)](#)

[MoveLeft\(\)](#)

[MovePageDn\(\)](#)

[MovePageUp\(\)](#)

[MovePanEnd\(\)](#)

[MovePanHome\(\)](#)

[MovePanLeft\(\)](#)

[MovePanRight\(\)](#)

[MoveRight\(\)](#)

[MoveUp\(\)](#)

[Print\(\)](#)

[Rotate\(\)](#)

[TypeBar\(\)](#)

[TypeBarStock\(\)](#)

[TypeLine\(\)](#)

[TypePie\(\)](#)

[TypeStacked\(\)](#)

[TypeStackedPercent\(\)](#)

[VerticalScroll\(\)](#)

[VerticalScrollRefresh\(\)](#)

[ZoomIn\(\)](#)

[ZoomOut\(\)](#)

### **System Properties**

*These properties are used internally. They are provided as reference only and should NEVER be accessed directly in your applications.*

#### **Inherits From**

(No ancestors)

#### **Inherited By**

(No descendants)

## **GraphWindow:DataSource Export**

### **Description**

Reference to the data source object. Should be a type of data server.

### **Type**

OBJECT

## GraphWindow:Graph Export

### Description

Reference to the graph object.

### Type

OBJECT [GraphInWindow](#)

## GraphWindow:AutoLayout() Method

### Purpose

Automatically add all columns from the datasource to the graph.

### Syntax

```
<oGraphWindow>:( <oDataSource > ) ---> NIL
```

### Arguments

*<oDataSource>*                Reference to the data source object. Should be a type of data server.

### Returns

NIL

### Description

The first column is used to create the labels for the X Axis ( [AxisX:GetLabelArray](#) ) and each column or field name is used for the title ( [GraphColumn:Title](#) )

## **GraphWindow:TypeBar() Method**

### **Purpose**

Set all columns to bar graphs.

### **Syntax**

```
<oGraphWindow>.TypeBar() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Light Lib Business allows multiple graph types to be used by different columns in the same graph. For example, you can mix bar and line graph types in the same graph. Furthermore, each column maintains unique visual attributes such as the 3D and filled effects.

## **GraphWindow:TypeBarStock() Method**

### **Purpose**

Set all columns to stock graphs.

### **Syntax**

```
<oGraphWindow>.TypeBarStock() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Light Lib Business allows multiple graph types to be used by different columns in the same graph. For example, you can mix bar and line graph types in the same graph. Furthermore, each column maintains unique visual attributes such as the 3D and filled effects.



## **GraphWindow:Close() Method**

### **Purpose**

Close the window.

### **Syntax**

```
<oGraphWindow>:Close( <oEvent> ) ----> NIL
```

### **Arguments**

<oEvent>	Event object
----------	--------------

### **Returns**

NIL

## **GraphWindow:CloseGraph() Method**

### **Purpose**

Close a graph window

### **Syntax**

<oGraphWindow>.CloseGraph() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:DecAngleZToX() Method**

### **Purpose**

Decreases the Z Axis to X Axis angle and repaints the graph. The default value is 5.

### **Syntax**

<oGraphWindow>:DecAngleZToX() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:DecYToX() Method**

### **Purpose**

Decreases the Y Axis to X Axis ratio and repaints the graph. The default value is 10.

### **Syntax**

<oGraphWindow>:DecYToX() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:DecZToX() Method**

### **Purpose**

Decreases the Z Axis to X Axis ratio and repaints the graph. The default value is 10.

### **Syntax**

<oGraphWindow>:DecZToX() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## GraphWindow:DefaultAspect() Method

### Purpose

Reset [GraphInWindow:RatioAutoYToX](#), [GraphInWindow:RatioZToX](#) and [GraphInWindow:AngleZtoX](#) values to default values.

### Syntax

```
<oGraphWindow>:DefaultAspect() ---> NIL
```

### Arguments

None

### Returns

NIL

## **GraphWindow:Destroy() Method**

### **Purpose**

Destroy the graph object and reference to the datasource.

### **Syntax**

```
<oGraphWindow>:Destroy() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

## GraphWindow:Effect3D() Method

### Purpose

Toggle the [GraphInWindow:Effect3D](#) and redisplay the graph.

### Syntax

```
<oGraphWindow>:Effect3D() ---> NIL
```

### Arguments

None

### Returns

NIL



## **GraphWindow:EffectFilled() Method**

### **Purpose**

Toggle the [GraphInWindow:EffectFilled](#) and redisplay the graph.

### **Syntax**

```
<oGraphWindow>:EffectFilled() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:Expose() Method**

### **Purpose**

Paint the graph in the window.

### **Syntax**

```
<oGraphWindow>:Expose( <oEvent> ) ---> NIL
```

### **Arguments**

<oEvent>	Event object
----------	--------------

### **Returns**

NIL

## **GraphWindow:FileExit() Method**

### **Purpose**

Close the file being graphed.

### **Syntax**

<oGraphWindow>.FileExit() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:HorizontalScroll() Method**

### **Purpose**

Activate the Horizontal scroll bar object.

### **Syntax**

<oGraphWindow>.HorizontalScroll( <oScrollEvent > ) ---> NIL

### **Arguments**

<oScrollEvent>            Scroll object.

### **Returns**

NIL

## **GraphWindow:HorizontalScrollRefresh() Method**

### **Purpose**

Refresh the Horizontal scroll bar. See [GraphWindow:HorizontalScroll\(\)](#)

### **Syntax**

<oGraphWindow>:HorizontalScrollRefresh() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## GraphWindow:IncAngleZToX() Method

### Purpose

Increment the angle between the Z and X axis.

### Syntax

```
<oGraphWindow>:IncAngleZToX( <nPercent> ) ---> NIL
```

### Arguments

<nPercent>	Percent to increase
------------	---------------------

### Returns

NIL

## **GraphWindow:IncYToX() Method**

### **Purpose**

Increment the angle between the Y and X axis.

### **Syntax**

<oGraphWindow>:IncYToX( <nPercent> ) ---> NIL

### **Arguments**

<nPercent>                      Percent to increase.

### **Returns**

NIL

## **GraphWindow:IncZToX() Method**

### **Purpose**

Increment the angle between the Z and X axis.

### **Syntax**

<oGraphWindow>:IncZToX( <*nPercent*> ) ---> NIL

### **Arguments**

<*nPercent*>                      Percent to increase.

### **Returns**

NIL



## GraphWindow:Init() Method

### Purpose

Create a graph window capable of displaying a [GraphInWindow](#) object.

### Syntax

```
GraphWindow{ <oOwner>, <oDataSource> } ----> SELF
```

### Arguments

<oOwner>	Owner object
<oDataSource>	Default data source to be used by all columns. Should be a type of data server.

### Returns

SELF	A reference to a new GraphWindow object
------	---

## **GraphWindow:TypeLine() Method**

### **Purpose**

Change the graph type to pie.

### **Syntax**

```
<oGraphWindow>.TypeLine() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Light Lib Business allows multiple graph types to be used by different columns in the same graph. For example, you can mix bar and line graph types in the same graph. Furthermore, each column maintains unique visual attributes such as the 3D and filled effects.

## **GraphWindow:MouseDown() Method**

### **Purpose**

Activate the down button on the mouse.

### **Syntax**

<oGraphWindow>:MouseDown( <oMouseEvent> ) ---> NIL

### **Arguments**

<oMouseEvent>            Mouse event object

### **Returns**

NIL

## **GraphWindow:Print() Method**

### **Purpose**

Print the graph

### **Syntax**

```
<oGraphWindow>.Print() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

When you choose to print a graph, you will receive the standard Windows Print dialog window.

## **GraphWindow:TypePie() Method**

### **Purpose**

Set all columns to be bar graphs.

### **Syntax**

```
<oGraphWindow>.TypePie() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Each pie represents a column and each slice represents a value from the data source.

This type of graph does not allow mixing column graph types.

## **GraphWindow:Rotate() Method**

### **Purpose**

Rotate the graph's skipper. See [GraphInWindow:SkipperRotate.](#)

### **Syntax**

<oGraphWindow>:Rotate() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:TypeStacked() Method**

### **Purpose**

Change the graph type to stacked.

### **Syntax**

```
<oGraphWindow>.TypeStacked() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Each bar represents a column and each portion of the column represents a value from the data source. The total of all the values in a column is represented by the height of the bar.

This type of graph does not allow mixing column graph types.

## **GraphWindow:TypeStackedPercent() Method**

### **Purpose**

Set the graph to be a stacked percentage bar graph. Each column is the same size.

### **Syntax**

```
<oGraphWindow>:TypeStackedPercent() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Each bar represents a column and each portion of the column represents a percentage of the total of all values.

This type of graph does not allow mixing column graph types.



## **GraphWindow:VerticalScroll() Method**

### **Purpose**

Activate the Vertical scroll bar object.

### **Syntax**

```
<oGraphWindow>.VerticalScroll( <oScrollEvent> ) ---> NIL
```

### **Arguments**

<i>&lt;oScrollEvent&gt;</i>	Scroll event object
-----------------------------	---------------------

### **Returns**

NIL

## **GraphWindow:VerticalScrollRefresh() Method**

### **Purpose**

Refresh the Vertical scroll bar. See [GraphWindow:VerticalScroll\(\)](#)

### **Syntax**

<oGraphWindow>.VerticalScrollRefresh() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:MoveDn() Method**

### **Purpose**

Move to the next record or row in the data source.

### **Syntax**

<oGraphWindow>.MoveDn() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:MoveGoBottom() Method**

### **Purpose**

Move to the bottom or end of the data source.

### **Syntax**

<oGraphWindow>.MoveGoBottom() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:MoveGoTop() Method**

### **Purpose**

Move to the top or beginning of the data source.

### **Syntax**

<oGraphWindow>.MoveGoTop() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:MoveLeft() Method**

### **Purpose**

Pan the columns in view left.

### **Syntax**

<oGraphWindow>:MoveLeft() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:MovePageDn() Method**

### **Purpose**

Move one page down in the data source.

### **Syntax**

<oGraphWindow>.MovePageDn() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:MovePageUp() Method**

### **Purpose**

Move one page up in the data source.

### **Syntax**

<oGraphWindow>.MovePageUp() ---> NIL

### **Arguments**

None

### **Returns**

NIL



## **GraphWindow:MovePanEnd() Method**

### **Purpose**

Pan to the last column in the graph.

### **Syntax**

<oGraphWindow>.MovePanEnd() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:MovePanHome() Method**

### **Purpose**

Pan to the first column in the graph.

### **Syntax**

<oGraphWindow>.MovePanHome() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:MovePanLeft() Method**

### **Purpose**

Pan one view of columns left.

### **Syntax**

<oGraphWindow>.MovePanLeft() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:MovePanRight() Method**

### **Purpose**

Pan one view of columns right.

### **Syntax**

<oGraphWindow>.MovePanRight() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:MoveRight() Method**

### **Purpose**

Pan the columns in view right.

### **Syntax**

<oGraphWindow>.MoveRight() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphWindow:MoveUp() Method**

### **Purpose**

Move to the previous record or row in the data source.

### **Syntax**

```
<oGraphWindow>.MoveUp() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

## GraphWindow:ZoomIn() Method

### Purpose

Zoom in the graph. Display less rows and columns.

### Syntax

<oGraphWindow>:ZoomIn( <nRow>, <nCol>, <nRowMin>, <nColMin> ) ---> NIL

### Arguments

<nRow>	Number of rows to remove from the view
<nCol>	Number of columns to remove from the view
<nRowMin>	Minimum number of rows to be in the view
<nColMin>	Minimum number of columns to be in the view

### Returns

NIL

## GraphWindow:ZoomOut() Method

### Purpose

Zoom out from the graph. Display more rows and columns.

### Syntax

<oGraphWindow>:ZoomOut( <nRow>, <nCol>, <nRowMin>, <nColMin> ) ---> NIL

### Arguments

<nRow>	Number of rows to add to the view
<nCol>	Number of columns to add to the view
<nRowMin>	Minimum number of rows to be in the view
<nColMin>	Minimum number of columns to be in the view

### Returns

NIL



## GraphInWindow Class

### Purpose

Provide a graph capable of being displayed inside a window.

### Properties

[AngleZToX Access/Assign](#)  
[AxisX Export](#)  
[AxisYLeft Export](#)  
[AxisYRight Export](#)  
[ColorBack Access/Assign](#)  
[ColorColumnFreeze Access/Assign](#)  
[CoordinateBottom Access/Assign](#)  
[CoordinateHeight Access/Assign](#)  
[CoordinateLeft Access/Assign](#)  
[CoordinateMaximize Access/Assign](#)  
[CoordinateRight Access/Assign](#)  
[CoordinateTop Access/Assign](#)  
[CoordinateWidth Access/Assign](#)  
[DataSource Export](#)  
[Effect3D Access/Assign](#)  
[EffectFilled Access/Assign](#)  
[EraseText Access/Assign](#)  
[FirstColInView Access/Assign](#)  
[hDC Access/Assign](#)  
[hWnd Access/Assign](#)  
[Legend Export](#)  
[MarginBottom Access/Assign](#)  
[MarginLeft Access/Assign](#)  
[MarginRight Access/Assign](#)  
[MarginTop Access/Assign](#)  
[NumColInGraph Access/Assign](#)  
[NumColInView Access/Assign](#)  
[NumFirstColInView Access/Assign](#)  
[NumRowInView Access/Assign](#)  
[OnError Access/Assign](#)  
[OnMouse Access/Assign](#)  
[RatioAutoYToX Access/Assign](#)  
[RatioYToX Access/Assign](#)  
[SideXYColorFill Access/Assign](#)  
[SideXYEffect3D Access/Assign](#)  
[SideXYGrid Access/Assign](#)  
[SideXYTransparent Access/Assign](#)  
[SideZXColorFill Access/Assign](#)  
[SideZXEffect3D Access/Assign](#)  
[SideZXGrid Access/Assign](#)  
[SideZXTransparent Access/Assign](#)  
[SideZYColorFill Access/Assign](#)  
[SideZYEfect3D Access/Assign](#)  
[SideZYGrid Access/Assign](#)  
[SideZYTransparent Access/Assign](#)  
[SkipperRotate Access/Assign](#)  
[Title Access/Assign](#)  
[TitleFont Export](#)

## WndBackColor Access/Assign

### **Methods**

[AddColumn\(\)](#)  
[AddColumnDouble\(\)](#)  
[AddColumnQuad\(\)](#)  
[AddColumnTriple\(\)](#)  
[ColorWheel\(\)](#)  
[Destroy\(\)](#)  
[Display\(\)](#)  
[ExecOnMouse\(\)](#)  
[ForceStable\(\)](#)  
[GetColumnInGraph\(\)](#)  
[GetColumnInView\(\)](#)  
[Init\(\)](#)  
[Invalidate\(\)](#)  
[IsTypeBar\(\)](#)  
[IsTypeBarStock\(\)](#)  
[IsTypeLine\(\)](#)  
[IsTypePie\(\)](#)  
[IsTypeStacked\(\)](#)  
[IsTypeStackedPercent\(\)](#)  
[MoveBottom\(\)](#)  
[MoveDn\(\)](#)  
[MoveLeft\(\)](#)  
[MovePageDn\(\)](#)  
[MovePageUp\(\)](#)  
[MovePanEnd\(\)](#)  
[MovePanHome\(\)](#)  
[MovePanLeft\(\)](#)  
[MovePanRight\(\)](#)  
[MoveRight\(\)](#)  
[MoveTop\(\)](#)  
[MoveUp\(\)](#)  
[Print\(\)](#)  
[Stabilize\(\)](#)  
[StabilizeAll\(\)](#)  
[TypeBar\(\)](#)  
[TypeBarStock\(\)](#)  
[TypeLine\(\)](#)  
[TypePie\(\)](#)  
[TypeResetAll\(\)](#)  
[TypeStacked\(\)](#)  
[TypeStackedPercent\(\)](#)  
[ZoomIn\(\)](#)  
[ZoomOut\(\)](#)

### **System Properties**

*These properties are used internally. They are provided as reference only and should NEVER be accessed directly in your applications.*

[cbOnError Export](#)  
[cbOnMouse Export](#)  
[!UseClassSkipper Export](#)  
[oGraph Export](#)

### **Inherits From**

(No ancestors)

**Inherited By**

(No descendants)

## **GraphInWindow:AngleZToX Access/Assign**

### **Description**

Get/Set the Z to X angle. Range is 0 - 89 degrees. Zero means perpendicular to the X Axis, which in effect causes the graph to become 2D by forcing the ZY side to disappear.

### **Type**

INT

## GraphInWindow:AxisX Export

### Description

Reference to the AxisX object contained in this graph.

### Type

OBJECT [AxisX](#)

## GraphInWindow:AxisYLeft Export

### Description

Reference to the AxisYLeft object contained in this graph.

### Type

OBJECT [AxisYLeft](#)

## GraphInWindow:AxisYRight Export

### Description

Reference to the AxisYRight object contained in this graph..

### Type

OBJECT [AxisYRight](#)

## GraphInWindow:cbOnError Export

### Description

OnError codeblock. Do not assign a value directly, instead use [GraphInWindow:OnError](#)

### Type

CODEBLOCK



## GraphInWindow:cbOnMouse Export

### Description

OnMouse codeblock. Do not assign a value directly, instead use [GraphInWindow:OnMouse](#)

### Type

CODEBLOCK

## **GraphInWindow:ColorBack Access/Assign**

### **Description**

Get/Set the color used to display the graph background. The default is the [GraphInWindow:WndBackColor](#) color (the window background color).

### **Type**

OBJECT Color

## **GraphInWindow:ColorColumnFreeze Access/Assign**

### **Description**

Get/Set the color to be used to display all frozen columns in the graph.

### **Type**

OBJECT Color

## **GraphInWindow:CoordinateBottom Access/Assign**

### **Description**

Get/Set the bottom coordinate of the graph.

### **Type**

INT

## **GraphInWindow:CoordinateHeight Access/Assign**

### **Description**

Get/Set the height of the graph.

### **Type**

INT

## **GraphInWindow:CoordinateLeft Access/Assign**

### **Description**

Get/Set the left coordinate of the graph.

### **Type**

INT

## **GraphInWindow:CoordinateMaximize Access/Assign**

### **Description**

Get/Set if the coordinates of the graph of window are maximized. If coordinates are not specified, LLB will use the [GraphInWindow:hDC](#) or [GraphInWindow:hWnd](#) . However, if any of the coordinates are assigned CoordinateMaximize is set to FALSE. The only way to switch back to the entire device context is to set GraphInWindow:CoordinateMaximize to TRUE.

### **Type**

LOGICAL

## **GraphInWindow:CoordinateRight Access/Assign**

### **Description**

Get/Set the right coordinate of the graph.

### **Type**

INT



## **GraphInWindow:CoordinateTop Access/Assign**

### **Description**

Get/Set the top coordinate of the graph.

### **Type**

INT

## **GraphInWindow:CoordinateWidth Access/Assign**

### **Description**

Get/Set the width of the graph.

### **Type**

INT

## **GraphInWindow:dwSelf Export**

### **Description**

Reference to the GraphInWindow object (self) in the DLL.

### **Type**

DWORD

## **GraphInWindow:Effect3D Access/Assign**

### **Description**

Logical flag which determines if the graph columns have a default three dimensional (3D) appearance. This can be overridden by individual column object within the graph.

### **Type**

LOGICAL

## **GraphInWindow:EffectFilled Access/Assign**

### **Description**

Logical flag which determines if the graph columns are filled in. This can be overridden by each individual column object within the graph.

### **Type**

LOGICAL

## **GraphInWindow:EraseText Access/Assign**

### **Description**

Logical flag which determines if text in the graph is to be erased upon graph stabilization.

If autosizing is TRUE, when the graph rescales the Y Axis, the old Major Step labels will need to be erased. If EraseText is TRUE, the WndBackColor will be used to erase these Major Step labels. If EraseText is FALSE, you would need to explicitly erase that region.

If FALSE, the background is not erase. It is assumed that the background has already been cleared. For example, the FALSE setting would prevent the destruction of any possible BMP in the background.

### **Type**

LOGICAL

## **GraphInWindow:FirstColInView Access/Assign**

### **Description**

Get/Set a reference to the first column in the view.

### **Type**

INT

## **GraphInWindow:hDC Access/Assign**

### **Description**

Get/Set the device context value.

### **Type**

PTR



## **GraphInWindow:hWnd Access/Assign**

### **Description**

Get/Set the window handle value.

### **Type**

PTR

## GraphInWindow:Legend Export

### Description

Reference to the legend object contained in this graph.

### Type

OBJECT [Legend](#)

## **GraphInWindow:IUseClassSkipper Export**

### **Description**

Logical flag which determines if the class skipper is to be used.

### **Type**

LOGICAL

## **GraphInWindow:MarginBottom Access/Assign**

### **Description**

Get/Set the bottom margin of the graph within the window. The margin is a percentage of the window height.

### **Type**

DWORD

## **GraphInWindow:MarginLeft Access/Assign**

### **Description**

Get/Set the left margin of the graph within the window. The margin is a percentage of the window width.

### **Type**

DWORD

## **GraphInWindow:MarginRight Access/Assign**

### **Description**

Get/Set the right margin of the graph within the window. The margin is a percentage of the window width.

### **Type**

DWORD

## **GraphInWindow:MarginTop Access/Assign**

### **Description**

Get/Set the top margin of the graph within the window. The margin is a percentage of the window height.

### **Type**

DWORD

## GraphInWindow:NumCollnGraph Access

### Description

Get the number of columns in the graph.

### Type

INT



## **GraphInWindow:NumCollnView Access/Assign**

### **Description**

Get/Set the number of columns in the current view. This is used to display more (or less) columns in a graph.

### **Type**

INT

## **GraphInWindow:NumFirstCollInView Access/Assign**

### **Description**

Get/Set the absolute number of the first column in the view. For example, the first column in view may be the fifth column in the graph.

### **Type**

INT

## **GraphInWindow:NumRowInView Access/Assign**

### **Description**

Get/Set the number of rows to be viewed. This is used to display more (or less) rows in a graph.

### **Type**

INT

## **GraphInWindow:oDataSource Export**

### **Description**

Reference to the data source object. Should be a type of data server.

### **Type**

OBJECT

## GraphInWindow:oGraph Export

### Description

Reference to the Light Lib Business Graph object.

### Type

OBJECT [GraphInWindow](#)

## **GraphInWindow:OnError Access/Assign**

### **Description**

Get/Set a codeblock which is evaluated when an error occurs on the Graph.

### **Type**

CODEBLOCK

## **GraphInWindow:OnMouse Access/Assign**

### **Description**

Get/Set a codeblock which is evaluated when a mouse event occurs on the graph. See [OnMouse Codeblocks](#) .

### **Type**

CODEBLOCK

## **GraphInWindow:RatioAutoYToX Access/Assign**

### **Description**

Logical flag to determine if the YX ratio is to be automatically calculated.

This Ratio is the proportional percentage between the lengths of the Y and X axis. If the RatioAutoYToX feature is activated, then this ratio is affected by the window size. Light Lib Business determines the best ratio based on the window or [GraphInWindow:hDC](#) dimensions.

### **Type**

LOGICAL



## **GraphInWindow:RatioYToX Access/Assign**

### **Description**

Get/Set the YX ratio.

This ratio is the proportional percentage between the lengths of the Y and X axis. If the [GraphInWindow:RatioAutoYToX](#) is TRUE, then this ratio is affected by the window size. This ratio is between the width of the X Axis and the height of the Y Axis. The range is 10-200. 100 forces a square graph.

### **Type**

DWORD

## **GraphInWindow:RatioZToX Access/Assign**

### **Description**

Get/Set the ZX ratio.

This ratio is the proportional percentage between the lengths of the Z and X axis.

### **Type**

DWORD

## **GraphInWindow:SideXYColorFill Access/Assign**

### **Description**

Color used to display the graph's XY side.

### **Type**

OBJECT Color

## **GraphInWindow:SideXYEffect3D Access/Assign**

### **Description**

Logical flag to determine if the XY side has a 3D appearance.

### **Type**

LOGICAL

## **GraphInWindow:SideXYGrid Access/Assign**

### **Description**

Logical flag to determine if a grid is displayed on the XY side. The grid is based on the intersection of rows, columns and major steps.

### **Type**

LOGICAL

## **GraphInWindow:SideXYTransparent Access/Assign**

### **Description**

Logical flag to determine if the XY side is transparent.

### **Type**

LOGICAL

## **GraphInWindow:SideZXColorFill Access/Assign**

### **Description**

Color used to display the graph's ZX side.

### **Type**

OBJECT Color

## **GraphInWindow:SideZXEfFect3D Access/Assign**

### **Description**

Logical flag to determine if the ZX side has a 3D appearance.

### **Type**

LOGICAL



## **GraphInWindow:SideZXGrid Access/Assign**

### **Description**

Logical flag to determine if a grid is displayed on the ZX side. The grid is based on the intersection of rows and columns.

### **Type**

LOGICAL

## **GraphInWindow:SideZXTransparent Access/Assign**

### **Description**

Logical flag to determine if the ZX side is transparent.

### **Type**

LOGICAL

## **GraphInWindow:SideZYColorFill Access/Assign**

### **Description**

Color used to display the graph's ZY side.

### **Type**

OBJECT Color

## **GraphInWindow:SideZYEfect3D Access/Assign**

### **Description**

Logical flag to determine if the ZY side has a 3D appearance.

### **Type**

LOGICAL

## **GraphInWindow:SideZYGrid Access/Assign**

### **Description**

Logical flag to determine if a grid is displayed on the ZY side. The grid is based on the intersection of rows, columns and Major Steps.

### **Type**

LOGICAL

## **GraphInWindow:SideZYTransparent Access/Assign**

### **Description**

Logical flag to determine if the ZY side is transparent.

### **Type**

LOGICAL

## **GraphInWindow:SkipperRotate Access/Assign**

### **Description**

Logical flag to determine if the data skipper is rotated. By default the skipper is applied to rows. When the skipper is rotated, the skipper is applied to columns. In other words the Left() method behaves like a Dn() and Dn() behaves like Left() . This is useful when the data layout doesn't lend itself to being displayed in a conventional row/column relationship, but rather in a column/row relationship.

### **Type**

LOGICAL

**GraphInWindow:Title Access/Assign**

**Description**

Get/Set the graph title.

**Type**

STRING



## GraphInWindow:TitleFont Export

### Description

Font used to display the graph's title.

### Type

OBJECT [BaseFont](#)

## **GraphInWindow:WndBackColor Access/Assign**

### **Description**

Get/Set the graph background color. This is the background color used by all text in the graph. The window containing the graph is not affected by this color.

When you display a graph with AXIS when browsing : clear the entire window instead repaint a small section because its faster

### **Type**

OBJECT Color

## GraphInWindow:AddColumn() Method

### Purpose

Create a column object and add it to the graph.

### Syntax

`<oGraphInWindow>.AddColumn( <cbGetValue>, <siGetValue>, <oDataSource> ) ---> <oNewColumn>`

### Arguments

<code>&lt;cbGetValue&gt;</code>	Codeblock that receives two parameters (siGetValue and oDataSource) at eval time .
<code>&lt;siGetValue&gt;</code>	The field or column number
<code>&lt;oDataSource&gt;</code>	Column's data source

### Returns

<code>&lt;oNewColumn&gt;</code>	Reference to the column object
---------------------------------	--------------------------------

### Description

Once the graph is created, use this AddColumn() method to add column objects to the graph. See [Sample Graph](#)

## GraphInWindow:AddColumnDouble() Method

### Purpose

Create a column object and add it to the graph.

### Syntax

```
<oGraphInWindow>:AddColumnDouble( <cbGetValue1>, <siGetValue1>, <cbGetValue2>,  
<siGetValue2>, <oDataSource> ) ---> <oNewColumn>
```

### Arguments

<cbGetValue1>	Codeblock that receives two parameters (siGetValue1 and oDataSource) at eval time .
<siGetValue1>	The field or column number
<cbGetValue2>	Codeblock that receives two parameters (<siGetValue2> and <oDataSource>) at eval time .
<siGetValue2>	The field or column number
<oDataSource>	Column's data source

### Returns

<oNewColumn>Reference to the column object

### Description

Once the graph is created, use this AddColumn() method to add column objects to the graph.

## GraphInWindow:AddColumnQuad() Method

### Purpose

Create a column object and add it to the graph.

### Syntax

```
<oGraphInWindow>.AddColumnQuad( <cbGetValue1>, <siGetValue1>, <cbGetValue2>, <siGetValue2>,  
<cbGetValue3>, <siGetValue3>, <cbGetValue4>, <siGetValue4>, <oDataSource> ) ----> <oNewColumn>
```

### Arguments

<cbGetValue1>	Codeblock that receives two parameters (siGetValue1 and oDataSource) at eval time
<siGetValue1>	The field or column number
<cbGetValue2>	Codeblock that receives two parameters (siGetValue2 and oDataSource) at eval time
<siGetValue2>	The field or column number
<cbGetValue3>	Codeblock that receives two parameters (siGetValue3 and oDataSource) at eval time
<siGetValue3>	The field or column number
<cbGetValue4>	Codeblock that receives two parameters (siGetValue4 and oDataSource) at eval time
<siGetValue4>	The field or column number
<oDataSource>	Column's data source

### Returns

<oNewColumn>Reference to the column object

### Description

Once the graph is created, use this AddColumnQuad() method to add column objects to the graph.

## GraphInWindow:AddColumnTriple() Method

### Purpose

Create a column object and add it to the graph.

### Syntax

```
<oGraphInWindow>:AddColumnTriple( <cbGetValue1>, <siGetValue1>, <cbGetValue2>, <siGetValue2>,  
<cbGetValue3>, <siGetValue3>, <oDataSource> ) ---> <oNewColumn>
```

### Arguments

<cbGetValue1>	Codeblock that receives two parameters (siGetValue1 and oDataSource) at eval time
<siGetValue1>	The field or column number
<cbGetValue2>	Codeblock that receives two parameters (siGetValue2 and oDataSource) at eval time
<siGetValue2>	The field or column number
<cbGetValue3>	Codeblock that receives two parameters (siGetValue3 and oDataSource) at eval time
<siGetValue3>	The field or column number
<oDataSource>	Column's data source

### Returns

<oNewColumn>Reference to the column object

### Description

Once the graph is created, use this AddColumnTriple() method to add column objects to the graph.

## **GraphInWindow:Destroy() Method**

### **Purpose**

Destroy the GraphInWindow object.

### **Syntax**

<oGraphInWindow>:Destroy() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

Free all resources allocated to this object. This method is called automatically when the containing object is destroyed. You do not need to call this method directly.

## GraphInWindow:ColorWheel() Method

### Purpose

Set the *nColor*'th color of the color wheel to *oNewColor*

### Syntax

`<oGraphInWindow>:ColorWheel( <nColor>, <oNewColor> ) ---> NIL`

### Arguments

`<nColor>`                      Color wheel position to assign

`<oNewColor>`                  Color object

### Returns

NIL

### Description

The color wheel is used to automatically select a color for each new column added to the graph. The color wheel supports up to 32 different colors.



## GraphInWindow:Display() Method

### Purpose

Display the graph inside the window. If [GraphInWindow:hDc](#) is equal to zero, the method assumes the use of the window handle, otherwise it will use the Device Context (hDc)

### Syntax

```
<oGraphInWindow>:Display( <oWindow>, <hDc> ) ---> NIL
```

### Arguments

<oWindow>	Window containing the Graph object
<hDc>	Device Context handle.

### Returns

NIL

## **GraphInWindow:MoveDn() Method**

### **Purpose**

Move the skipper down to the next record or row.

### **Syntax**

<oGraphInWindow>:MoveDn() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## GraphInWindow:ExecOnMouse() Method

### Purpose

Pass the mouse event to the DLL which executes a callback function if an object belonging to a Light Lib Business graph, including the graph, was clicked on.

### Syntax

```
<oGraphInWindow>:ExecOnMouse( <oMouseEvent>, <oWindow> ) ---> NIL
```

### Arguments

<oMouseEvent>           MouseEvent object.

<oWindow>               Window containing the graph object.

### Returns

NIL

### Description

oWindow and oMouseEvent are passed to the DLL to provide references to the objects for the callback function to use.

## **GraphInWindow:ForceStable() Method**

### **Purpose**

Force the graph to become stable.

### **Syntax**

```
<oGraphInWindow>:ForceStable() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

## GraphInWindow:GetColumnInGraph() Method

### Purpose

Return the nColNum'th column object in the graph

### Syntax

<oGraphInWindow>:GetColumnInGraph( <nColNum> ) ---> <oColumn>

### Arguments

<nColNum>                      Column to get.

### Returns

<oColumn>                      Column object

## GraphInWindow:GetColumnInView() Method

### Purpose

Return the nColNum'th column object in the view

### Syntax

<oGraphInWindow>:GetColumnInView( <nColNum> ) ---> <oColumn>

### Arguments

<nColNum>	Column to get.
-----------	----------------

### Returns

<oColumn>	Column object
-----------	---------------

## **GraphInWindow:MoveBottom() Method**

### **Purpose**

Move the skipper to the bottom or end of the data source.

### **Syntax**

<oGraphInWindow>:MoveBottom() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphInWindow:MoveTop() Method**

### **Purpose**

Move the skipper to the top or beginning of the data source.

### **Syntax**

```
<oGraphInWindow>:MoveTop() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL



## **GraphInWindow:TypeBar() Method**

### **Purpose**

Set all columns to be bar graphs.

### **Syntax**

```
<oGraphInWindow>:TypeBar() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Light Lib Business allows multiple graph types to be used by different columns in the same graph. For example, you can mix bar and line graph types in the same graph. Furthermore, each column maintains unique visual attributes such as the 3D and filled effects.

## **GraphInWindow:TypeBarStock() Method**

### **Purpose**

Set all columns to be bar stock graphs.

### **Syntax**

```
<oGraphInWindow>.TypeBarStock() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Light Lib Business allows multiple graph types to be used by different columns in the same graph. For example, you can mix bar and line graph types in the same graph. Furthermore, each column maintains unique visual attributes such as the 3D and filled effects.

## **GraphInWindow:TypeLine() Method**

### **Purpose**

Set all columns to be line graphs.

### **Syntax**

```
<oGraphInWindow>:TypeLine() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Light Lib Business allows multiple graph types to be used by different columns in the same graph. For example, you can mix bar and line graph types in the same graph. Furthermore, each column maintains unique visual attributes such as the 3D and filled effects.

## **GraphInWindow:TypePie() Method**

### **Purpose**

Change the graph type to pie.

### **Syntax**

```
<oGraphInWindow>:TypePie() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Each pie represents a column and each slice represents a value from the data source.

This type of graph does not allow mixing column graph types.

## **GraphInWindow:TypeResetAll() Method**

### **Purpose**

Reset all columns in the graph to be the graph type

### **Syntax**

```
<oGraphInWindow>:TypeResetAll() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

## **GraphInWindow:TypeStacked() Method**

### **Purpose**

Change the graph type to stacked.

### **Syntax**

```
<oGraphInWindow>.TypeStacked() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Each bar represents a column and each portion of the column represents a value from the data source. The total of all the values in a column is represented by the height of the bar.

This type of graph does not allow mixing column graph types.

## **GraphInWindow:TypeStackedPercent() Method**

### **Purpose**

Set the graph to be a stacked percentage bar graph. Each column is the same size.

### **Syntax**

```
<oGraphInWindow>.TypeStackedPercent() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

Each bar represents a column and each portion of the column represents a percentage of the total of all values.

This type of graph does not allow mixing column graph types.

## **GraphInWindow:Init() Method**

### **Purpose**

Create a graph inside the DLL. Initialize all graph sub-objects including legend, X and Y axis.

### **Syntax**

GraphInWindow{ <oDataSource> } ---> SELF

### **Arguments**

<oDataSource>            Default data source to be used by all columns. Should be a type of data server.

### **Returns**

SELF                      A reference to a new GraphInWindow object



## **GraphInWindow:Invalidate() Method**

### **Purpose**

Force the graph to be fully refreshed in the next stabilization cycle.

### **Syntax**

```
<oGraphInWindow>:Invalidate() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

## GraphInWindow:IsTypeBar() Method

### Purpose

Returns if the graph is a bar graph.

### Syntax

<oGraphInWindow>:IsTypeBar() ---> </Value>

### Arguments

None

### Returns

</Value>                      If the graph is a bar graph

## GraphInWindow:IsTypeBarStock() Method

### Purpose

Returns if the graph is a bar stock graph.

### Syntax

```
<oGraphInWindow>:IsGraphBarStock() ---> </Value>
```

### Arguments

None

### Returns

</Value>	If the graph is a bar stock graph
----------	-----------------------------------

## GraphInWindow:IsTypeLine() Method

### Purpose

Returns if the graph is a line graph

### Syntax

<oGraphInWindow>:IsTypeLine() ---> </Value>

### Arguments

None

### Returns

</Value>                      If the graph is a line graph

## GraphInWindow:IsTypePie() Method

### Purpose

Returns if the graph is a pie graph

### Syntax

```
<oGraphInWindow>:IsTypePie() ---> </Value>
```

### Arguments

None

### Returns

</Value>                      If the graph is a pie graph

## GraphInWindow:IsTypeStacked() Method

### Purpose

Returns if the graph is a stacked bar graph

### Syntax

```
<oGraphInWindow>.IsTypeStacked() ---> </Value>
```

### Arguments

None

### Returns

</Value>                      If the graph is a stacked bar graph

## GraphInWindow:IsTypeStackedPercent() Method

### Purpose

Returns if the graph is a stacked percentage bar graph

### Syntax

```
<oGraphInWindow>.IsTypeStacked() ---> </Value>
```

### Arguments

None

### Returns

</Value>                      If the graph is a stacked bar graph

## **GraphInWindow:MoveLeft() Method**

### **Purpose**

Pan the columns in view left.

### **Syntax**

```
<oGraphInWindow>:MoveLeft() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

This allows navigation through the data source columns. Frozen columns are respected and are not removed from view.



## **GraphInWindow:MovePageDn() Method**

### **Purpose**

Move the skipper one page down in the data source.

### **Syntax**

<oGraphInWindow>:MovePageDn() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphInWindow:MovePageUp() Method**

### **Purpose**

Move the skipper one page up in the data source.

### **Syntax**

<oGraphInWindow>.MovePageUp() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## **GraphInWindow:MovePanEnd() Method**

### **Purpose**

Pan to the last column in the graph.

### **Syntax**

```
<oGraphInWindow>:MovePanEnd() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

This allows navigation through the data source columns. Frozen columns are respected and are not removed from view.

## **GraphInWindow:MovePanHome() Method**

### **Purpose**

Pan to the first column in the graph.

### **Syntax**

<oGraphInWindow>:MovePanHome() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

This allows navigation through the data source columns. Frozen columns are respected and are not removed from view.

## **GraphInWindow:MovePanLeft() Method**

### **Purpose**

Pan one view of columns left.

### **Syntax**

<oGraphInWindow>:MovePanLeft() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

This allows navigation through the data source columns. Frozen columns are respected and are not removed from view.

## **GraphInWindow:MovePanRight() Method**

### **Purpose**

Pan one view of columns right.

### **Syntax**

<oGraphInWindow>:MovePanRight() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

This allows navigation through the data source columns. Frozen columns are respected and are not removed from view.

## **GraphInWindow:Print() Method**

### **Purpose**

Print the graph.

### **Syntax**

```
<oGraphInWindow>:Print() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

When you choose to print a graph, you will receive the standard Windows Print dialog window.

## **GraphInWindow:MoveRight() Method**

### **Purpose**

Pan one column right.

### **Syntax**

<oGraphInWindow>:MoveRight() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

This allows navigation through the data source columns. Frozen columns are respected and are not removed from view.



## GraphInWindow:Stabilize() Method

### Purpose

Incrementally stabilize the graph and return if the graph is stable.

### Syntax

```
<oGraphInWindow>:Stabilize() ---> </Stable>
```

### Arguments

None

### Returns

</Stable>                      If the graph is stable

### Description

Light Lib Business graphs follow an incremental stabilization process. For example, when a window is enlarged, the graph needs to be scaled and redisplayed but the data does not have to be refreshed from the data source. This incremental stabilization allows for excellent graphing performance.

## GraphInWindow:StabilizeAll() Method

### Purpose

Incrementally stabilize the entire graph and return if the graph is stable.

### Syntax

```
<oGraphInWindow>:Stabilize() ---> </Stable>
```

### Arguments

None

### Returns

</Stable>                      If the graph is stable

### Description

Light Lib Business graphs follow an incremental stabilization process. For example, when a window is enlarged, the graph needs to be scaled and redisplayed but the data does not have to be refreshed from the data source. This incremental stabilization allows for excellent graphing performance.

## **GraphInWindow:MoveUp() Method**

### **Purpose**

Move the skipper up one row or record in the data source.

### **Syntax**

<oGraphInWindow>:MoveUp() ---> NIL

### **Arguments**

None

### **Returns**

NIL

## GraphInWindow:ZoomIn() Method

### Purpose

Zoom in the graph. Display less rows and columns.

### Syntax

```
<oGraphInWindow>:ZoomIn( <nRow>, <nCol>, <nRowMin>, <nColMin> ) ----> NIL
```

### Arguments

<nRow>	Number of rows to remove from the view
<nCol>	Number of columns to remove from the view
<nRowMin>	Minimum number of rows to be in the view
<nColMin>	Minimum number of columns to be in the view

### Returns

NIL

## GraphInWindow:ZoomOut() Method

### Purpose

Zoom out from the graph. Display more rows and columns.

### Syntax

```
<oGraphInWindow>:ZoomOut( <nRow>, <nCol>, <nRowMin>, <nColMin> ) ---> NIL
```

### Arguments

<nRow>	Number of rows to add to the actual view
<nCol>	Number of columns to add to the actual view
<nRowMin>	Minimum number of rows to be in the view
<nColMin>	Minimum number of columns to be in the view

### Returns

NIL

## Legend Class

### Purpose

Contains legend information associated with the graph columns.

### Properties

[colorBack Access/Assign](#)  
[colorOutline Access/Assign](#)  
[OnError Access/Assign](#)  
[OnMouse Access/Assign](#)  
[Visible Access/Assign](#)

### Methods

[Destroy\(\)](#)  
[Init\(\)](#)  
[IsOnBest\(\)](#)  
[IsOnBottom\(\)](#)  
[IsOnLeft\(\)](#)  
[IsOnRight\(\)](#)  
[IsOnTop\(\)](#)  
[OnBest\(\)](#)  
[OnBottom\(\)](#)  
[OnLeft\(\)](#)  
[OnRight\(\)](#)  
[OnTop\(\)](#)

### System Properties

*These properties are used internally. They are provided as reference only and should NEVER be accessed directly in your applications.*

[cbOnError Export](#)  
[cbOnMouse Export](#)  
[dwSelf Export](#)  
[oGraph Export](#)

### Inherits From

(No ancestors)

### Inherited By

(No descendants)

## Legend:cbOnError Export

### Description

OnError codeblock. Do not assign a value directly, instead use [Legend:OnError](#)

### Type

CODEBLOCK

## Legend:cbOnMouse Export

### Description

OnMouse codeblock. Do not assign a value directly, instead use [Legend:OnMouse](#)

### Type

CODEBLOCK



**Legend:colorBack Access/Assign**

**Description**

Color object used to display the legend background.

**Type**

OBJECT

**Legend:colorOutline Access/Assign**

**Description**

Font and attributes Color used to display the legend outline.

**Type**

OBJECT

**Legend:dwSelf Export**

**Description**

Reference to the legend object (self) in the DLL.

**Type**

DWORD

## Legend: oGraph Export

### Description

Reference to the graph object.

### Type

OBJECT [GraphInWindow](#)

## **Legend:OnError Access/Assign**

### **Description**

Get/Set a codeblock which is evaluated when an error occurs on the legend.

### **Type**

CODEBLOCK

## **Legend:OnMouse Access/Assign**

### **Description**

Get/Set a codeblock which is evaluated when a mouse event occurs on the legend. See [OnMouse Codeblock](#) .

### **Type**

CODEBLOCK

**Legend:Visible Access/Assign**

**Description**

Logical flag to determine if the legend is visible.

**Type**

LOGICAL

## **Legend:Destroy() Method**

### **Purpose**

Destroy the legend object.

### **Syntax**

<oLegend>.Destroy() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

Free all resources allocated to this object. This method is called automatically when the containing object is destroyed. You do not need to call this method directly.



## Legend:Init() Method

### Purpose

Initialize a legend object.

### Syntax

Legend{ <oGraph> } ---> SELF

### Arguments

<oGraph>	Graph object
----------	--------------

### Returns

SELF	Reference to a new legend object
------	----------------------------------

## Legend:IsOnBest() Method

### Purpose

Returns if the legend is in the best position on the graph.

### Syntax

```
<oLegend>.IsOnBest() ---> </Value>
```

### Arguments

None

### Returns

</Value>                      If legend is in the best position in the graph

### Description

[Legend:OnBest\(\)](#) automatically displays the legend in the best position in the graph.

## Legend:IsOnBottom() Method

### Purpose

Returns if the legend is on the bottom of the graph.

### Syntax

```
<oLegend>.IsOnBottom() ---> </Value>
```

### Arguments

None

### Returns

</Value>                      If the legend is on the bottom of the graph

### Description

[Legend:OnBottom\(\)](#) displays the legend on the bottom of the graph.

## Legend:IsOnLeft() Method

### Purpose

Returns if the legend is on the left side of the graph.

### Syntax

```
<oLegend>.IsOnLeft() ---> </Value>
```

### Arguments

None

### Returns

</Value>                      If the legend is on the left side of the graph

### Description

[Legend:OnLeft\(\)](#) displays the legend on the left side of the graph.

## Legend:IsOnRight() Method

### Purpose

Returns if the legend is on the right side of the graph.

### Syntax

```
<oLegend>:IsOnRight() ---> </Value>
```

### Arguments

None

### Returns

</Value>                      If the legend is on the right side of the graph

### Description

[Legend:OnRight\(\)](#) displays the legend on the right side of the graph.

## Legend:IsOnTop() Method

### Purpose

Returns if the legend is on top of the graph.

### Syntax

```
<oLegend>.IsOnTop() ---> </Value>
```

### Arguments

None

### Returns

</Value>                      If the legend is on top of the graph

### Description

[Legend:OnTop\(\)](#) displays the legend on top of the graph.

## **Legend:OnBest() Method**

### **Purpose**

Automatically display the legend in the best position in the graph.

### **Syntax**

<oLegend>:OnBest() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

[Legend:IsOnBest\(\)](#) returns if this feature is enabled.

## **Legend:OnBottom() Method**

### **Purpose**

Set the legend to be displayed on the bottom of the graph

### **Syntax**

```
<oLegend>.OnBottom() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

[Legend:IsOnBottom\(\)](#) returns if this feature is enabled.



## **Legend:OnLeft() Method**

### **Purpose**

Set the legend to be displayed on the left side of the graph

### **Syntax**

<oLegend>.OnLeft() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

Legend:IsOnLeft() returns if this feature is enabled.

## **Legend:OnRight() Method**

### **Purpose**

Set the legend to be displayed on the right side of the graph

### **Syntax**

<oLegend>.OnRight() ---> NIL

### **Arguments**

None

### **Returns**

NIL

### **Description**

[Legend:IsOnRight\(\)](#) returns if this feature is enabled.

## **Legend:OnTop() Method**

### **Purpose**

Set the legend to be displayed on the top of the graph

### **Syntax**

```
<oLegend>.OnTop() ---> NIL
```

### **Arguments**

None

### **Returns**

NIL

### **Description**

[Legend:IsOnTop\(\)](#) returns if this feature is enabled.

## Simple Graph Sample

The following is sample code which creates a simple GraphInWindow Object. This can be used to create a graph with Light Lib Business.

```
//*****
METHOD DoOpenArrayGraph(aDataArray) CLASS GraphWindow

Local aValues      As Array
Local nI           As ShortInt
Local oColumn      As Object

aValues := { {"January", -3,-5,6,8,7,10,60,70,80,90 }      ,;
             {"February",2,-5,11,5,2,-51,61,71,81,91 }    ,;
             {"March",-4,7,10,6,12,-52,62,72,82,92 }      ,;
             {"April", 5,2,5,7,4,-53,63,73,83,93 }        ,;
             {"May",12,17,27,37,47,-57,67,77,87,9 }      ,;
             {"June",15,18,28,38,48,-58,68,78,88,98},,;
             {"July",9,19,29,39,49,-59,69,79,89,99 } }

// Open an Array DataServer
Self:DataSource := ArrayDataSource{ aDataArray }

// Create a New Business graph inside a window with
// a reference to a DataServer
Self:Graph := GraphInWindow{ Self:DataSource }

// Use the first column as X labels
Self:Graph:AxisX:GetLabelArray := { { |siParam,oDataSource|
                                     asString( oDataSource:FieldGet( siParam )
                                     ) }, 1, Self:DataSource }

Self:Graph:AxisX:Title := "Title for the X Axis"

// Add all the other columns to the graph
For nI := 2To Self:DataSource:FCount

    // CA-Visual Objects does not support detached locals.
    // LLB allows you to store an integer which will be
    // passed to the code block when evaluated, also you
    // can define as a third parameter
    // a DataSource different from the Graph DataSource, if needed
    oColumn := Self:Graph:AddColumn( { |siParam,oDataSource|
                                     oDataSource:FieldGet(siParam) } ,nI,
                                     Self:DataSource )

    // Add a title to the column
    oColumn:TitleRight := Self:DataSource:FieldName(nI)

Next nI

// Set the graph title
Self:Graph:Title := "Simple Light Lib Business Graph"

Self:Repaint()
```

## Complex Graph Sample

The following is sample code which creates a complex GraphInWindow Object. This can be used to create a graph with Light Lib Business.

```
//*****
METHOD DoOpenArrayGraph(aDataArray) CLASS GraphWindow

Local aValues      As Array
Local nI           As ShortInt
Local oColumn      As Object

aValues := { {"January", -3,-5,6,8,7,10,60,70,80,90 }      ,;
             {"February",2,-5,11,5,2,-51,61,71,81,91 }    ,;
             {"March",-4,7,10,6,12,-52,62,72,82,92 }      ,;
             {"April", 5,2,5,7,4,-53,63,73,83,93 }        ,;
             {"May",12,17,27,37,47,-57,67,77,87,9 }      ,;
             {"June",15,18,28,38,48,-58,68,78,88,98 }    ,;
             {"July",9,19,29,39,49,-59,69,79,89,99 } }

// Open an Array DataServer
Self:DataSource := ArrayDataSource{ aDataArray }

// Create a New Business graph inside a window with
// a reference to a DataServer
Self:Graph := GraphInWindow{ Self:DataSource }

// Use the first column as X labels
Self:Graph:AxisX:GetLabelArray := {{ |siParam,oDataSource|
                                   asString( oDataSource:FieldGet( siParam )
                                   ) }, 1, Self:DataSource }

Self:Graph:AxisX:Title := "Title for the X Axis"

// Add all the other columns to the graph
For nI := 2To Self:DataSource:FCount

    // CA-Visual Objects does not support detached locals.
    // LLB allows you to store an integer which will be passed
    // to the code block when evaluated, also you can define
    // as a third parameter a DataSource different from
    // the Graph DataSource, if needed
    oColumn := Self:Graph:AddColumn( { |siParam,oDataSource|
                                     oDataSource:FieldGet(siParam)} , nI,
                                     Self:DataSource )

    // Add a title to the column
    oColumn:TitleRight := Self:DataSource:FieldName(nI)

    If nI==3

        // This syntax illustrate how to create a # OnMouse
        // for each column of the graph
        oColumn:OnMouse := { |oLLB,liSubLLB,r8Data,oWindow,oMouseEvent|
                            MyOnMouse(oLLB,liSubLLB,r8Data,oWindow,oMouseEvent) }
```

```

        // This syntax illustrate how to dimension/position a bar
        oColumn:WidthX      := 50
        oColumn:WidthY      := 50
        oColumn:OffsetX     := -20
        oColumn:OffsetY     := -40

        // This syntax is useful when creating line axis
        oColumn:PenWidth := 3

        // This syntax show how to make a column # than graph
        // This is Only for PRO edition
        oColumn:ColumnTypeLine()
        oColumn:Effect3d     := TRUE
        oColumn:EffectFilled := FALSE

        // This is Only for PRO edition
        oColumn:AttachedToLeftAxis := FALSE
        oColumn:TitleRight := "TitleRight2"

        oColumn:TitleFont:Name      := "ARIAL"
oColumn:TitleFont:Bold      := TRUE
        oColumn:TitleFont:Color     := Color{255,255,255}

        // This syntax illustrates how to configure all the colors
        oColumn:ColorFill           := Color{0,255,0} // Blue
        oColumn:ColorFillShadow     := Color{0,0,255} //Green
        oColumn:Color               := Color{255,0,0} // Red

        // This syntax allow to freeze a column at the beginning
        oColumn:Freeze := TRUE

    Endif

Next nI

Self:Graph:AxisYLeft:Title := "YAxisLeftTitle"

// This section is a sample to assign access value for Y Axis
Self:Graph:AxisYLeft:Min := -100
Self:Graph:AxisYLeft:Max := 500
Self:Graph:AxisYLeft:Base:= -20

// Sample of how to access an axis value
r8Temp := Self:Graph:AxisYLeft:Max

// Adjust the Major and Minor steps
self:Graph:AxisYLeft:StepMinor := 10
self:Graph:AxisYLeft:StepMajor := 100

r8Temp := Self:Graph:AxisYLeft:StepMinor
r8Temp := Self:Graph:AxisYLeft:StepMajor

Self:Graph:AxisYLeft:Header := "YAxisLeftHeader"

// This illustrates how to transform the Axis Y scale values
Self:Graph:AxisYLeft:ValueToString := { |r8Data| Transform(r8Data,"99999.99")
}

```

```

// This illustrates how to use the ValueToScale feature
Self:Graph:AxisYLeft:ValueToScale := { |r8Data| r8Data*1000 }

// This illustrates how to create a # OnMouse for AxisYLeft
Self:Graph:AxisYLeft:OnMouse := { |oLLB,liSubLLB,r8Data,oWindow,oMouseEvent|
MyOnMouse(oLLB,liSubLLB,r8Data,oWindow,oMouseEvent) }

// This illustrates how to configure the sides colors
Self:Graph:SideXYColorFill := Color{0,255,0} // Blue
Self:Graph:SideZYColorFill := Color{0,0,255} // Green
Self:Graph:SideZXColorFill := Color{255,0,0} // Red

Self:Graph:AxisYLeft:TitleFont:Bold := TRUE
Self:Graph:AxisYLeft:TitleFont:Underline := TRUE

Self:Graph:AxisYLeft:HeaderFont:Name := "ARIAL"
Self:Graph:AxisYLeft:HeaderFont:Underline := TRUE
Self:Graph:AxisYLeft:HeaderFont:Color := Color{0,255,0}

Self:Graph:AxisYLeft:LabelFont:Name := "ARIAL"
Self:Graph:AxisYLeft:LabelFont:Italic := TRUE
Self:Graph:AxisYLeft:LabelFont:Color := Color{255,0,0}

Self:Graph:AxisYRight:LabelFont:Name := "ARIAL"
Self:Graph:AxisYRight:LabelFont:Bold := TRUE
Self:Graph:AxisYRight:LabelFont:Color := Color{0,0,255}

// Define the legend as visible and allow automatic best
// positioning
Self:Graph:Legend:Visible := TRUE
Self:Graph:Legend:OnBest()

// This demonstrate the legend color options
Self:Graph:Legend:ColorBack := Color{0,0,255}
Self:Graph:Legend:ColorOutLine := Color{255,0,255}

// Set proportion Ratios to defaults
Self:Graph:RatioAutoYToX := True

// Set the proportion ratio to a value
Self:Graph:RatioY2X := 10
Self:Graph:RatioZToX := 38
Self:Graph:AngleZToX := 38

// Set the graph title
Self:Graph:Title := "Sample"
Self:Graph:TitleFont:Name := "WIDE LATIN"
Self:Graph:TitleFont:Bold := TRUE
Self:Graph:TitleFont:Italic := TRUE
Self:Graph:TitleFont:Color := Color{0,255,255}

Self:Repaint()

```

## **MS-Visual Basic**

No help is available at this time.



## **MS-Visual Basic Functions & Classes**

## **C/C++**

No help is available at this time.

## **C/C++ Functions & Classes**

## Common Problems

[Unable to load a DLL at runtime](#)

[DLL Crashes](#)

[Out of Memory](#)

## Tips & Techniques

## **Editions**

### **Light Lib Business**

Features of the Standard edition of Light Lib Business:

Dynamic Navigation	End-users can move through a data source and the data values are automatically graphed.
Automatic Dialogs	Default end-user dialog windows for every element in a graph.
Automatic Scaling	All graph elements are automatically kept in proportion even when large values come into view along side smaller values.
Automatic Sizing	A graph is automatically sized according to its bounding window coordinates.
Column Positioning	Complete control over the size and offset of each column.

### **Light Lib Business Pro**

In addition to the standard features, the Pro edition provides the following additional powerful graphing features:

BLOB Support	Save a graph and all its settings to disk for later retrieval.
Mixed Graphs	Display and maintain different graph types (eg. line or bar) for each column in a graph.
Complex Graphs	Graph financial/stock data hiliting minimum and maximum values.
Left and Right Y Axis	Attach columns to either the Left or the Right Y Axis.
Rotate Skipper	Change or flip both the graph and direction of the skipper.

## **bMove()**

## **DLL Functions**

### **Purpose**

Move or shift columns in a graph.

### **Syntax**

```
bMove(      dwLLBusiness AS DWORD,  
            liDirection   AS LONGINT,  
            siRepeat      AS SHORTINT ) ---> liError
```

### **Arguments**

*dwLLBusiness*            A pointer to an existing graph

*liDirection*            Direction to move the graph columns. One of the following is valid:

```
LLB_MOVE_RIGHT  
LLB_MOVE_LEFT  
LLB_MOVE_PAN_RIGHT  
LLB_MOVE_PAN_LEFT  
LLB_MOVE_PAN_HOME  
LLB_MOVE_PAN_END  
LLB_MOVE_UP  
LLB_MOVE_DOWN  
LLB_MOVE_PAGE_UP  
LLB_MOVE_PAGE_DOWN  
LLB_MOVE_GO_TOP  
LLB_MOVE_GO_BOTTOM
```

*siRepeat*                Number of times to move the column.

### **Returns**

*liError*                An error status value.

### **Description**

This function relies on the Skipper/GoTop/GoBottom [callback](#) functions.

## bOnMouse()

## DLL Functions

### Purpose

Process the mouse event.

### Syntax

```
bOnMouse(    dwLLBusiness AS DWORD,  
             siMouseX     AS SHORTINT,  
             siMouseY     AS SHORTINT,  
             oWindow      AS OBJECT,  
             oMouseEvent  AS OBJECT,  
             dwExtraParam AS DWORD ) ---> NIL
```

### Arguments

<i>dwLLBusiness</i>	Pointer to the graph object
<i>siMouseX</i> , <i>siMouseY</i>	The X and Y coordinates of the mouse in pixels.
<i>oWindow</i>	Window object
<i>oMouseEvent</i>	Mouse event object
<i>dwExtraParam</i>	This parameter is passed to your Idle callback function. It should be a pointer to any kind of structure.

### Description

If the mouse event happens on the graph object or on any of the graph's sub-objects, the mouse [callback](#) function associated with the selected object will be called.

### Example

See [ExecOnMouse\(\)](#)



## **bPrint()**

## **DLL Functions**

### **Purpose**

Print a graph.

### **Syntax**

```
bPrint(dwLLBusiness AS DWORD,  
hDCPrinter AS SHORTINT,  
liScaleRatio AS LONGINT,  
siX1 AS SHORTINT,  
siY1 AS SHORTINT,  
siX2 AS SHORTINT,  
siY2 AS SHORTINT ) ---> liError
```

### **Arguments**

<i>dwLLBusiness</i>	Pointer to the graph object
<i>hDCPrinter</i>	Printer device context. If 0 is used, the user is prompted with the default windows printer selection dialog.
<i>liScaleRatio</i>	Ratio used to set the printer density with the display density. The value 1000 is a 1:1 ratio.
<i>siX1</i> , <i>siY1</i> , <i>siX2</i> , <i>siY2</i>	Coordinates used to print the graph. These are used to define the size of the graph on the destination printer page. If <i>siX1</i> is -1, then the graph coordinates are used to print the graph.

### **Returns**

*liError* Error value. 0 indicates success.

### **Description**

This allows a graph to be printed.

## **bStabilize()**

## **DLL Functions**

### **Purpose**

Stabilize a graph object

### **Syntax**

```
bStabilize(    dwLLBusiness AS PTR,  
              siStabilizeAll AS SHORTINT) ---> liState
```

### **Arguments**

*dwLLBusiness*            A pointer to a graph object

*siStabilizeAll*            Type of stabilization

### **Returns**

*liState*                    This returns one of the following values

LLB\_GRAPH\_STATE\_INVALID  
LLB\_GRAPH\_STATE\_REFRESH\_DATA  
LLB\_GRAPH\_STATE\_BUILD  
LLB\_GRAPH\_STATE\_DISPLAY  
LLB\_GRAPH\_STATE\_STABLE

### **Description**

bStabilize() is called until LLB\_GRAPH\_STATE\_STABLE is returned. A return value of LLB\_GRAPH\_STATE\_INVALID indicates that stabilization is not possible due to an error with a passed parameter. In this situation, the error [callback](#) function will have been called first.

## **Color2RGB()**

## **CA-Visual Objects**

### **Purpose**

Convert a Color to Red Green Blue format

### **Syntax**

Color2RGB( oColor AS OBJECT ) ----> liColor

### **Arguments**

oColor A color object.

### **Returns**

liColor The color value of the color object's Red, Green and Blue values.

## IsGraph()

## CA-Visual Objects

### Purpose

Determines if the passed object is a graph

### Syntax

IsGraph( *xLLB* ) ---> *IValue*

### Arguments

*xLLB*                      An OBJECT or DWORD pointing to any Light Lib Business Object

### Returns

*IValue*                      If the passed object is nil, FALSE is returned.

### Description

This function is polymorphic. It can receive a CA-Visual Objects object pointing to any Light Lib Business Object, or a DWORD to any Light Lib Business object.

## **IsGraphAxisLeftY()**      **CA-Visual Objects**

### **Purpose**

Determines if the passed object is the LeftYAxis.

### **Syntax**

IsGraphAxisLeftY(      *xLLB* ) ---> *IValue*

### **Arguments**

*xLLB*                      An OBJECT or DWORD pointing to any Light Lib Business Object

### **Returns**

*IValue*                      If the passed object is nil, FALSE is returned.

### **Description**

This function is polymorphic. It can receive a CA-Visual Objects object pointing to any Light Lib Business Object, or a DWORD to any Light Lib Business object.

## **IsGraphAxisRightY()**      **CA-Visual Objects**

### **Purpose**

Determines if the passed object is the Right Y Axis.

### **Syntax**

IsGraphAxisLeftY(      xLLB ) ---> IValue

### **Arguments**

*xLLB*                      An OBJECT or DWORD pointing to any Light Lib Business Object

### **Returns**

*IValue*                      If the passed object is nil, FALSE is returned.

### **Description**

This function is polymorphic. It can receive a CA-Visual Objects object pointing to any Light Lib Business Object, or a DWORD to any Light Lib Business object.

## IsGraphAxisX()

## CA-Visual Objects

### Purpose

Determines if the passed object is the X Axis.

### Syntax

IsGraphAxisX( *xLLB* ) ---> *IValue*

### Arguments

*xLLB*                      An OBJECT or DWORD pointing to any Light Lib Business Object

### Returns

*IValue*                      If the passed object is nil, FALSE is returned.

### Description

This function is polymorphic. It can receive a CA-Visual Objects object pointing to any Light Lib Business Object, or a DWORD to any Light Lib Business object.

## **IsGraphColumn()**                      **CA-Visual Objects**

### **Purpose**

Determines if the passed object is a column.

### **Syntax**

IsGraphColumn( *xLLB* ) ---> *IValue*

### **Arguments**

*xLLB*                                      An OBJECT or DWORD pointing to any Light Lib Business Object

### **Returns**

*IValue*                                      If the passed object is nil, FALSE is returned.

### **Description**

This function is polymorphic. It can receive a CA-Visual Objects object pointing to any Light Lib Business Object, or a DWORD to any Light Lib Business object.



## **IsGraphLegend()**                      **CA-Visual Objects**

### **Purpose**

Determines if the passed object is a legend

### **Syntax**

IsGraphLegend( *xLLB* ) ---> *IValue*

### **Arguments**

*xLLB*                                      An OBJECT or DWORD pointing to any Light Lib Business Object

### **Returns**

*IValue*                                      If the passed object is nil, FALSE is returned.

### **Description**

This function is polymorphic. It can receive a CA-Visual Objects object pointing to any Light Lib Business Object, or a DWORD to any Light Lib Business object.

## LightLibBusinessPath() CA-Visual Objects

### Purpose

Get or set the path to the Light Lib Business installed directory

### Syntax

LightLibBusinessPath( *cNewPath* AS STRING ) ---> *cPath*

### Arguments

<i>cNewPath</i>	New path to set.
-----------------	------------------

### Returns

<i>cPath</i>	Current path
--------------	--------------

## ProcGoBottom()                      CA-Visual Objects

### Purpose

Move to the bottom of the data source.

### Syntax

ProcGoBottom(     *dwGraph*    AS DWORD ) ----> *siValue* Callback

### Arguments

<i>dwGraph</i>	Reference to the graph object
----------------	-------------------------------

### Returns

<i>siValue</i>	Error value
----------------	-------------

## ProcGoTop()

## CA-Visual Objects

### Purpose

Move to the top of the data source.

### Syntax

ProcGoTop( *dwGraph* AS DWORD ) ----> *siValue* Callback

### Arguments

<i>dwGraph</i>	Reference to the graph object
----------------	-------------------------------

### Returns

<i>siValue</i>	Error value
----------------	-------------

## ProcOnMouse()                      CA-Visual Objects

### Purpose

Default mouse event handler routine.

### Syntax

```
ProcOnMouse( dwLLBusiness AS DWORD,  
             liSubObject   AS LONGINT,  
             r8Data AS REAL8,  
             oWindow      AS OBJECT,  
             oMouseEvent AS OBJECT,  
             dwVoidParam AS DWORD ) ---> NIL Callback
```

### Arguments

<i>dwLLBusiness</i>	Reference to the registered application
<i>liSubObject</i>	Graph sub-object selected
<i>r8Data</i>	
<i>oWindow</i>	Window object containing the graph
<i>oMouseEvent</i>	Mouse event object
<i>dwVoidParam</i>	Optional parameters

### Returns

NIL

## StdMouseHandler()      CA-Visual Objects

### Purpose

The default mouse handling system.

### Syntax

```
StdMouseHandler(    oLLB            AS OBJECT,  
                    liSubLLB       AS LONGINT,  
                    r8Data AS REAL8,  
                    oWindow       AS OBJECT,  
                    oMouseEvent AS OBJECT ) ---> IHandled
```

### Arguments

<i>oLLB</i>	Light Lib Business object (Graph, Legend, Axis etc.)
<i>liSubLLB</i>	Possible Light Lib Business subobject (Legend's Font etc.)
<i>r8Data</i>	Real8 data, which is only applicable to some subobjects
<i>oWindow</i>	Window object that the graph object belongs to.
<i>oMouseEvent</i>	CA-Visual Objects MouseEvent

### Returns

<i>IHandled</i>	If the mouse event was handled
-----------------	--------------------------------

## **pszProcGetLabel()**      **CA-Visual Objects**

### **Purpose**

To retrieve the Label for the X Axis

### **Syntax**

`pszProcGetLabel( dwAxisX AS DWORD ) ----> pszLabelString` Callback

### **Arguments**

*dwAxisX*                      Pointer to the X Axis

### **Returns**

*pszLabelString*              The Label for the X Axis

## **siProcSkipper()**                      **CA-Visual Objects**

### **Purpose**

Skip or navigate through the data source.

### **Syntax**

```
siProcSkipper( dwGraph            AS DWORD,  
              nLineToSkip        AS SHORTINT ) ----> siLineToSkip CallBack
```

### **Arguments**

*dwGraph*                      Pointer to a graph

*siLineToSkip*                Number of rows to skip

### **Returns**

*siLineToSkip*                Number of rows skipped





**Who is DFL?** With corporate offices in Toronto, Paris and a research and development center in the south of France, DFL is fast becoming a leading developer of advanced add-on products for Windows and DOS. We are committed to providing the very best tools for serious software development. Let DFL's *Light Lib* family of products help you develop better applications.

Thank you for your support,  
**The DFL Team.**

## Light Lib Library User License

**This document is a contract between you (the licensee) and DFL.**

DFL supplies the software and grants a license for its use. You are totally responsible for choosing to use the software for the desired purpose, as well as for installing and using it, and for the results obtained.

DFL grants no rights for the software other than those explicitly stipulated in this agreement, and reserves all rights not explicitly granted to the licensee.

### License

DFL grants to the licensee a non-exclusive and non-transferable right to use the software for an unlimited duration. The licensee may make ONE copy of the software in readable form on a computer or other support for backup purposes.

You have the right to load the software into RAM and use it on a single computer, to install it on the hard disk of the computer, to produce executable files (.EXE) containing the Light Lib software.

You are not required to pay royalties to DFL on the sale of any applications using a Light Lib product. If you expect to sell or distribute more than 500 copies of a commercial product which uses one or more Light Lib products, it is required that you inform DFL and that you grant DFL permission to publicize the fact that your product uses Light Lib technology.

### Forbidden

The following are forbidden: sharing of the software on a network: each programmer that uses Light Lib software in part or in whole, must possess his or her own registered license, distribution of the software, decompilation of supplied files, disassembly of supplied files, rewriting and any form of reverse engineering of the software, and any other action or activity involving the software that is not explicitly authorized.

### Term

The license shall remain in force until it is cancelled. The licensee may cancel it at any time by destroying the software and all copies. It is also cancelled without notification if the licensee violates any terms or conditions of the present agreement. The licensee agrees to destroy the software and all copies if the agreement is cancelled.

### Limited warranty

The software is furnished as is, with no warranty of any sort, explicit or implicit, including, but not restricted to, implicit warranties as to its fitness or sale for any particular use. Any risks stemming from the quality and performance of the program are entirely borne by the licensee. If there is any defect in the program, the licensee (and not DFL or any intermediary) will assume any expenses for help, repair, or correction.

DFL does not guarantee that the functions included in the programs correspond to the needs of the licensee or that the program will function without interruptions or errors.

Nevertheless, DFL warrants that the diskettes on which the program is supplied are without defects in material and packaging for a period of ninety (90) days, to be counted from the date of delivery to the licensee as indicated on the bill of sale.

The only responsibility of DFL and the only compensation due to the licensee are:

1) Replacement of any diskette not in conformity with the limited warranty of DFL and which has been returned to DFL or an intermediary authorized by DFL, with a copy of the licensee's receipt, or,

2) If DFL or an authorized intermediary is unable to supply a diskette free of material or packaging defects the licensee may cancel this agreement by returning the software, and the licensee will be reimbursed.

In no case will DFL be responsible to the licensee for damages of any kind, including loss of profits or savings or direct or indirect loss resulting from use of or inability to use the program, even if DFL or an authorized intermediary of DFL was informed of the possibility of such damages, or of any claim for any third party.

### **General considerations**

You certify that you have read and understood the present agreement, and that you agree to be bound by its terms and conditions. You also recognize that it constitutes the sole and exclusive basis for our contract, replacing any earlier proposal or contract, verbal or written, and any other communication between us relating to the object of the present agreement.

### **Registered trademarks**

All trademarks are the property of their registered owners.

## Technical Support

The three primary means of technical support are via CompuServe, FAX and on our BBS. If it is an emergency, you can call or fax us.

### North America

DFL Software Inc.	Voice (416) 789-2223
1712 Avenue Road	Fax (416) 789-0204
Box 54616	BBS (416) 784-9712
Toronto, ON, M5M 4N5	CompuServe 74723,3321
CANADA	Internet 74723.3321@compuserve.com

### Europe

DFL Europe	Voice (33 1) 46 05 20 66
39-41, rue de la Saussière	Fax (33 1) 46 04 10 39
Boulogne	BBS (33 1) 46 05 26 88
FRANCE	CompuServe 100067,652
	Internet 100067.652@compuserve.com



## products by DFL

All Light Lib products have been designed and developed to be implemented easily and execute quickly .

**Windows**      Light Lib Business  
                      Light Lib Images  
                      Light Lib Multimedia

**DOS**            Light Lib Business  
                      Light Lib Images  
                      Light Lib Graphics

**Light Lib Business** is a revolutionary graphing library. It provides the unprecedented power to present users with "live" graphs. Your users will now be able to dynamically scroll and interact with graph data as if they were scrolling text data. The days of static graphs are over!

**Light Lib Images** is the most comprehensive image and document managing library available. Scanning, loading, saving, printing images or documents has never been easier.

**Light Lib Multimedia** is the easiest-to-use multimedia library for Windows. Adding the ability to play or record sound and display video, will bring your applications to new heights.

**Light Lib Graphics** for CA-Clipper is the first Replaceable Terminal Driver (RTD) for CA-Clipper. It will immediately transforms your text mode applications into graphic mode.

All Light Lib products for DOS are upward compatible with their Windows counterpart. Each product comes with complete help files and source code to the extensive support functions and classes.

All Light Lib products for CA-Clipper are fully compatible with Real and Protected mode linkers (Exospace, Blinker and Causeway) and each product is fully integrated with CA-Clipper's VMM system.

## Light Lib Objects Functions

### DLL Functions

[oAccess\(\)](#)

[oAssign\(\)](#)

[oNew\(\)](#)

[oDel\(\)](#)

### Constants

[Constants](#)

## Light Lib Objects (LLO)

Light Lib Objects is not another Light Lib product. LLO manages memory allocation and the proper creation and deletion of all objects within the Light Lib DLLs themselves. Every Light Lib product for Windows relies on this support DLL. Please review the specific language implementation carefully because the usage of LLO differs slightly from language to language.

LLO provides object oriented technology to languages that do not support object oriented programming and provides enhanced features to languages that support OOP. In addition to standard OOP features such as inheritance, polymorphism, and encapsulation, LLO implements advanced OOP concepts such as inheriting from an owner class which is not the immediate parent, dynamic class creation, BLOB aggregation and much more. The following is an example:

ABSTRACT Class - GRAPH Class

ABSTRACT Class - COLUMN Class

There is no relationship between the GRAPH Class and the COLUMN Class. However, if a method or property is not available in an instance of the COLUMN Class, LLO will not use the ABSTRACT parent class definition, which is how OOP systems work today. Instead, LLO is able to use the class Owner's definition which could, for example, be a GRAPH.

## How Do I?

## CA-Visual Objects

### Register and unregister an application

You need to call [dwLightLibAppRegister\(\)](#) at the start of your program. This allows the Light Lib DLLs to be properly initialized. If this registration is not executed, you will receive errors.

At the end of execution, you will need to unregister your application with the Light Lib DLLs by calling [dwLightLibAppUnRegister\(\)](#).



## Light Lib Objects Constants

[Abstract](#)

[Application](#)

[Class](#)

[Error](#)

## Class Constants

LLO_CLASS_ABSTRACT	Abstract Class (Hidden)
LLO_CLASS_APPLICATION	Application Class
LLO_CLASS_CONTEXT	Context Class (Hidden)
LLO_CLASS_ERROR	Error Class

## **Abstract Constants**

LLO\_ABSTRACT\_APPLICATION  
LLO\_ABSTRACT\_CARGO  
LLO\_ABSTRACT\_CARGO\_COUNT  
LLO\_ABSTRACT\_CLASS\_ID  
LLO\_ABSTRACT\_CLASS\_NAME  
LLO\_ABSTRACT\_CLASS\_VERSION  
LLO\_ABSTRACT\_ERROR  
LLO\_ABSTRACT\_LIBRARY\_ID  
LLO\_ABSTRACT\_LIBRARY\_NAME  
LLO\_ABSTRACT\_LIBRARY\_VERSION  
LLO\_ABSTRACT\_OWNER

## **Application Constants**

LLO\_APPLICATION\_CARGO\_COUNT\_DEFAULT  
LLO\_APPLICATION\_CONTEXT  
LLO\_APPLICATION\_HANDLE  
LLO\_APPLICATION\_NAME

## Error Constants

### Error Class

LLO_ERROR_ACTION	
LLO_ERROR_OBJECT	
LLO_ERROR_MESSAGE	Error message
LLO_ERROR_NUMBER	
LLO_ERROR_PARAM	Extended depending on Error Type
LLO_ERROR_PROPERTY	Property define#
LLO_ERROR_PROPERTY_NAME	Property name

### LLO\_ERROR\_NUMBER

LLO\_ERROR\_CARGO\_OUT\_OF\_LIMIT  
LLO\_ERROR\_INVALID\_CLASS\_DEFINE  
LLO\_ERROR\_INVALID\_OWNER\_TYPE  
LLO\_ERROR\_INVALID\_PARAMETERS  
LLO\_ERROR\_INVALID\_ACCESS\_NEW  
LLO\_ERROR\_INVALID\_ACCESS\_DEL  
LLO\_ERROR\_INVALID\_ACCESS\_ACCESS  
LLO\_ERROR\_INVALID\_ACCESS\_ASSIGN  
LLO\_ERROR\_MEMORY\_ALLOCATION  
LLO\_ERROR\_NO\_ERROR  
LLO\_ERROR\_OBJECT\_ACCESS\_DENIED  
LLO\_ERROR\_OBJECT\_ASSIGN\_DENIED  
LLO\_ERROR\_READONLY\_PROPERTY  
LLO\_ERROR\_UNDEFINED\_PROPERTY

### LLO\_ERROR\_ACTION

LLO\_ACTION\_ACCESS  
LLO\_ACTION\_ASSIGN  
LLO\_ACTION\_DEL  
LLO\_ACTION\_NEW

## User Defined Constants

LLI_UDF_ABORT	User Defined Function Abort return value
LLI_UDF_CONT	User Defined Function Continue return value
LLI_UDF_ERROR	Error append during a Light Lib function execution
LLI_UDF_EXIT	Exit phase for a Light Lib function execution
LLI_UDF_IDLE	Idle phase for a Light Lib function execution
LLI_UDF_INIT	Init phase for a Light Lib function execution

## Overview

Light Lib Objects

## oAccess()

## DLL Functions

### Purpose

Access an object's instance variable. See also [Light Lib Objects](#)

### Syntax

```
oAccess(      dwLObject    AS DWORD,  
              dwProperty    AS DWORD,  
              dwExtraParam  AS DWORD ) ---> dwData
```

### Arguments

<i>dwLObject</i>	A Light Lib object.
<i>dwProperty</i>	A property belonging to this Light Lib object.
<i>dwExtraParam</i>	Used to access the LLI_IMAGE_CARGO value. For example, if LLI_IMAGE_CARGO is a structure, <i>dwExtraParam</i> would represent the byte offset into the structure.

### Returns

<i>dwData</i>	The value of the requested object member
---------------	--

### Description

*dwExtraParam* must be cast to DWORD. This allows the Light Lib DLL to pass a POINTER, SHORTINT, LONGINT etc.

### Examples

```
// This returns the name of the class to which the object belongs.  
oAccess( dwMyObject, LLO_ABSTRACT_CLASS_NAME, 0 )  
  
// This returns the value of the second cargo  
// instance variable for this object.  
oAccess( dwMyObject, LLO_ABSTRACT_CARGO, 2 )
```



## oAssign()

## DLL Functions

### Purpose

Assign any value to a defined variable of an object. See also [Light Lib Objects](#)

### Syntax

```
oAssign(      dwLObject    AS DWORD,  
              dwProperty    AS DWORD,  
              dwValue       AS DWORD,  
              dwExtraParam  AS DWORD ) ----> liError
```

### Arguments

<i>dwLObject</i>	A Light Lib object
<i>dwProperty</i>	The predefined value to change. You can only change or assign to the symbols noted as Assignable. You are not able to modify symbols that are Read Only symbols.
<i>dwValue</i>	The value to be assigned.
<i>dwExtraParam</i>	Used to access the LLI_IMAGE_CARGO value. For example, if LLI_IMAGE_CARGO is a structure, <i>dwExtraParam</i> would represent a byte offset into the structure.

### Returns

*liError*                      An error code.

### Description

The *dwExtraParam* and *dwValue* must be cast to DWORD. This allows the DLL to pass a POINTER, SHORINT, LONGINT etc.

### Examples

```
// This sets the cargo size for this object to 4 DWORD.  
oAssign(dwMyObject, LLO_ABSTRACT_CARGO_SIZE, 4 )  
  
//This sets the second cargo instance variable to dwMyValue.  
oAssign(dwMyObject, LLO_ABSTRACT_CARGO, dwMyValue, 2 )
```

## **oNew()**

## **DLL Functions**

### **Purpose**

Used to create a new Application Object. See also [Light Lib Objects](#)

### **Syntax**

```
oNew(      dwLLClass      AS DWORD,  
           dwLLObject     AS DWORD,  
           siSizeOfCargo  AS SHORTINT,  
           dwValue        AS DWORD  
           dwExtraParam   AS DWORD ) ---> ptrAppHnd
```

### **Arguments**

<i>dwLLClass</i>	Represents the class of the object to be created.
<i>dwLLObject</i>	Represents the object to be created. If the class to which the object belongs is an application, the <i>dwLLObject</i> doesn't need to be defined (pass zero).
<i>siSizeOfCargo</i>	The size or number of DWORD parameters in an object's cargo.
<i>dwValue</i>	This is an optional value containing extra information. For example, when you create a new Column object inside a Graph object, <i>dwValue</i> dictates where the column should be inserted. If <i>dwValue</i> is 0, the new column becomes the last column. If <i>dwValue</i> is an existing Column number, the new Column is inserted before the passed number.
<i>dwExtraParam</i>	An optional parameter.

### **Returns**

*dwAppHnd*                      A pointer to a Light Lib Objects application handle.

### **Description**

This allows you to register a Light Lib application with the Light Lib Objects DLL. This registration allows the Light Lib DLL to be used simultaneously by several applications in a multitasking operating system and to automate memory garbage collection. You must ensure that your applications always terminate with [oDel\(\)](#).

When you create an application that uses a Light Lib DLL, you need to register that application with the Light Lib Objects DLL. This needs to be done at the very start of your application by calling *oNew()* and by passing the proper arguments.

Once registered, Light Lib Objects, automatically keeps track of all objects created within the registered application. This guarantees that all objects are properly connected to the Light Lib DLL.

When terminating an application, you need to unregister it from the Light Lib Images DLL with *oDel()*. This frees all memory allocated to objects in the application, even if the objects have not been explicitly erased. It is, however, always better to erase images from memory when they are no longer needed using *oDel()*.

This *oNew()* and *oDel()* technique needs to be implemented to ensure that Light Lib Objects can properly manage all memory and processes when being called simultaneously from multiple applications. This is

very important in a multitasking operating system.

## **oDel()**

## **DLL Functions**

### **Purpose**

Delete any Light Lib object. This frees all memory allocated to objects in a registered Light Lib application. See also [Light Lib Objects](#)

### **Syntax**

`oDel(            dwLLObject        AS DWORD ) ---> dwAppHnd`

### **Arguments**

*dwLLObject*                      A DWORD representing any Light Lib object.

### **Returns**

*dwAppHnd*                        An empty pointer to a Light Lib Images application handle.

### **Description**

You must ensure that your Light Lib applications always terminate with oDel().

Be aware, that deleting a Light Lib object will also delete all of its child objects (if any) as well. As an example, deleting the Application object in turn deletes all objects created by that application from memory. It is highly recommended to delete the registered Application object by calling oDel() prior to exiting any Light Lib application.

This oNew() and oDel() technique (registering and unregistering) must be implemented to ensure that Light Lib Objects can properly manage the Light Lib DLLs when being called simultaneously from multiple applications. This is very important in a multitasking operating system.

## **dwLightLibApp()**

## **CA-Visual Objects**

### **Purpose**

Get the current Light Lib Application. See also [Light Lib Objects](#)

### **Syntax**

`dwLightLibApp()` ---> *dwLightLibRegisteredApp*

### **Arguments**

None.

### **Returns**

*dwLightLibRegisteredApp*      The registered application.

## **dwLightLibAppRegister()**

## **CA-Visual Objects**

### **Purpose**

Register this instance of application into the LLO.DLL This must be done only once in an application's execution, and prior to any calls to the Light Lib library you are using. See also [Light Lib Objects](#)

### **Syntax**

```
dwLightLibAppRegister(  oApp      AS OBJECT,  
                        oWindow    AS OBJECT ) ---> dwLightLibRegisteredApp
```

### **Arguments**

*oApp*                                      Application to register.

*oWindow*                                      Owner window.

### **Returns**

*dwLightLibRegisteredApp*              The value of the registered application.

### **Description**

This function is used to register your Light Lib application with Light Lib Objects. If this registration process is not done, your application will not work properly.

## **dwLightLibAppUnRegister()      CA-Visual Objects**

### **Purpose**

Unregister a Light Lib application from the Light Lib Objects DLL. See also [Light Lib Objects](#)

### **Syntax**

`dwLightLibAppUnRegister()` ---> *dwLightLibRegisteredApp*

### **Arguments**

None

### **Returns**

*dwLightLibRegisteredApp*      Unregister an application.

## Out of Memory

If you are experiencing memory problems in applications using Light Lib DLLs, there is a good chance that you are keeping unnecessary references to objects such as images or graphs in memory. When your application no longer needs an object, you should formally remove or delete it from memory by calling [oDel\(\)](#) with the proper parameters.



## DLL Crashes

This error could occur when multiple applications that use Light Lib DLLs are running simultaneously. In order to prevent conflicts between them, you must ensure that each application is registered with Light Lib Objects.

This involves making a call to [oNew\(\)](#) with the proper parameters at the beginning of your program. Also, remember to make a call to [oDel\(\)](#) just before your application terminates.

### **Unable to Load a DLL at Runtime**

Make sure that the proper Light Lib DLL is available in your WINDOWS\SYSTEM directory. At installation time, Light Lib DLLs are installed to this directory. If they are not present when your application runs, the applications will cause a LoadError().

