

Contents

User's Guide

Introduction

[Introducing List Pro](#)

[Information for Users of Previous Products](#)

[List Pro Controls](#)

[Using the FarPoint Property Designer](#)

[Tutorial](#)

[Using ActiveX Controls](#)

[Using DLL Controls](#)

How-to-Guides

[Getting Started](#)

[Working with Columns](#)

[Working with Groups](#)

[Working with List Items](#)

[Customizing the Control's Appearance](#)

Reference Guide

[Properties](#)

[Events](#)

[Functions and Methods](#)

[Messages \(DLL only\)](#)

[Structures \(DLL only\)](#)

[Styles \(DLL only\)](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

DataFieldList Property

[See Also](#)

Applies To

fpCombo control

Description

Sets or returns the data field name to which the list in an fpCombo control is bound. This property is available for Visual Basic users only.

Syntax

Visual Basic `[form.]fpCombo1.DataFieldList[= text$]`

Designer Page

[General subtab of the Data Binding designer page](#)

Remarks

The DataFieldList property lets you bind the list in a single-column fpCombo control to a different data field than the edit field. Use the DataField property to bind the edit field of an fpCombo control.

Use the DataFieldList property for single-column fpCombo controls. Do not set the DataFieldList property for a multiple-column fpCombo control; use the [ColDataField](#) property to bind specific columns.

For more information on how to bind a data field, refer to the DataField property in the Visual Basic documentation.

Data Type

String

See Also

[Creating Column Headers](#)

[Working with Databases](#)

[ColDataField](#), [DataSourceList](#) properties

PD	RD	WR	RT	DT
	✓	✓	✓	

DataSourcehWnd Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the window handle to a Data control on another form. This property is available at run time only. This property is available for VBX controls in Visual Basic only.

Syntax

Visual Basic `[form.]control.DataSourcehWnd[= value%]`

Remarks

The DataSourcehWnd property lets you bind a VBX control to a Data control located on another form. You can also use the DataSourcehWnd property to change to a different bound Data control.

To obtain the window handle of the Data control to which you want to bind the control, use the [ListPro_GetControlhWnd](#) function.

For all controls, before you set the DataSourcehWnd property, you must set the DataField property for the control to designate which field to display from the Data control.

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example creates two forms, one containing a two-column fpList control and one containing a two-column fpCombo control.

To try this example, create two forms. Place a Data control on Form1. Bind the Data control to Visual Basic's BIBLIO.MDB database using the DatabaseName property and to the Titles table using the RecordSource property. Place the fpList control on Form1 and bind it to the Data control using the DataSource property. The following code specifies that the fpList control displays the Title and Au_ID (Author ID) fields.

Create Form2 and place an fpCombo control on it. The following code binds the fpCombo control to the Data control on Form1 using the DataSourceWnd and [DataSourceWndList](#) properties. The fpCombo control displays the Title and Year Published fields.

When you run the project, click Form1 to display Form2.

Visual Basic

```
' Form Load event for Form1
Sub Form1_Load ()
fpList1.Columns = 2
fpList1.DataAutoSizeCols = False
fpList1.Col = 0
fpList1.ColDataField = "Title"
fpList1.ColWidth = 60
fpList1.Col = 1
fpList1.ColDataField = "Au_ID"
fpList1.ColWidth = 15
End Sub

' Show Form2 when user clicks Form1
Sub Form1_Click
Form2.Show
End Sub

' Form Load event for Form2
Sub Form2_Load ()
fpCombo1.Columns = 2
fpCombo1.DataAutoSizeCols = False
fpCombo1.ColumnEdit = 0
fpCombo1.Col = 0
fpCombo1.ColDataField = "Title"
fpCombo1.ColWidth = 60
fpCombo1.Col = 1
fpCombo1.ColDataField = "Year Published"
fpCombo1.ColWidth = 20
fpCombo1.DataSourceWnd = ListPro_GetControlhWnd(Form1.Data1)
fpCombo1.DataSourceWndList = ListPro_GetControlhWnd(Form1.Data1)
End Sub
```

See Also

[Binding to a Data Control on a Different Form](#)

[DataSourceWndList](#) property

[ListPro_GetControlhWnd](#) function

PD	RD	WR	RT	DT
	✓	✓	✓	

DataSourcehWndList Property

[See Also](#)

[Example](#)

Applies To

fpCombo control

Description

Sets or returns the window handle to a Data control on another form. This property is available at run time only. This property is available for VBX controls in Visual Basic only.

Syntax

Visual Basic `[form.]fpCombo1.DataSourcehWndList[= value%]`

Remarks

The DataSourcehWndList property lets you bind the list in an fpCombo control to a Data control located on another form. You can also use the DataSourcehWndList property to change to a different bound Data control.

To obtain the window handle of the Data control to which you want to bind the fpCombo list, use the [ListPro_GetControlhWnd](#) function.

Data Type

Integer

See Also

[Binding to a Data Control on a Different Form](#)

[DataSourceWnd](#) property

[ListPro_GetControlhWnd](#) function

PD	RD	WR	RT	DT
	✓		✓	✓

DataSourceList Property

[See Also](#)

[Example](#)

Applies To

fpCombo control

Description

Sets or returns the name of the Data control through which the list in an fpCombo control is bound. This property is read-only at run time. This property is available for Visual Basic users only.

Syntax

Visual Basic `[form.]fpCombo1.DataSourceList[= text$]`

Remarks

The DataSourceList property lets you bind the list in an fpCombo control to a different Data control than the edit field.

Double-click the DataSourceList property in the Properties window to bind the list to an available Data control.

Use the DataSource property to bind the edit field in an fpCombo control.

Data Type

String

See Also

[Working with Databases](#)

[DataFieldList](#) property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

DataSync Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the method for synchronizing the Data control and the selected item in the fpCombo or fpList controls. This property is available for Visual Basic users only.

Syntax

Visual Basic `[form.]control.DataSync[= setting%]`

Designer Page

[General subtab of the Data Binding designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - None	Removes synchronization between the selected list item and the Data control	LC_DATASYNC_NONE
1 - Update Data Control	Moves the Data control to a different record when the selected list item changes	LC_DATASYNC_UPDATE_DATA
2 - Update Bound Control	Changes the selected list item when the Data control moves to a different record	LC_DATASYNC_UPDATE_BOUND
3 - Update Both	(Default) Moves the Data control when the list item changes and selects another list item when the Data control moves	LC_DATASYNC_UPDATE_BOTH

When binding an fpCombo or fpList control to a database field, each value in the list represents a record. The DataSync property determines whether selecting a different list item also scrolls the Data control to another record and whether clicking the Data control's arrows also highlights another list item.

Data Type

Integer (Enumerated)

See Also

[Working with Databases](#)

[DataFieldList](#), [DataSourceList](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

EditHeight Property

[See Also](#) [Example](#)

Applies To

fpCombo control

Description

Sets or returns the height of the edit field in an fpCombo control.

Syntax

C	UINT CbxGetEditHeight(HWND hWnd, long FAR *lpValue); UINT CbxSetEditHeight(HWND hWnd, long value);
C++	long CfpComboBox::GetEditHeight(void); CfpComboBox::SetEditHeight(long value);
Visual Basic	[form.]fpCombo1.EditHeight[= value!]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

The default value for the EditHeight property is -1, which sets the height of the edit field to the size of the current default font.

In Visual Basic, the measurement unit used by the EditHeight property depends on the setting of the form's ScaleMode property. The default ScaleMode setting is twips (1/1440 of an inch). Generally the ActiveX and VBX controls use twips as the default measurement unit, and the DLL control uses pixels as the default measurement unit.

Data Type

Single

See Also

[MaxEditLen](#), [MultiLine](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

EnableKeyEvents Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether keyboard events occur for an fpCombo or fpList control. This property is available for ActiveX and VBX controls only.

Syntax

C++ **BOOL Class::GetEnableKeyEvents(void);**
 Class::SetEnableKeyEvents(BOOL value);

Visual Basic [form.]control.EnableKeyEvents[= boolean%]

Designer Page

[Miscellaneous designer page](#)

Remarks

The default value for the EnableKeyEvents property is True, which triggers the standard Visual Basic KeyDown, KeyPress, and KeyUp events. When the EnableKeyEvents property is set to False, the keyboard events do not occur.

Disabling these events speeds up the performance of the fpCombo and fpList controls.

Data Type

Integer (Boolean)

See Also

[EnableMouseEvents](#), [EnableTopChangeEvent](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

EnableMouseEvents Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether mouse events occur for an fpCombo or fpList control. This property is available for VBX controls only.

Syntax

C++ **BOOL Class::GetEnableMouseEvents(void);**
Class::SetEnableMouseEvents(BOOL value);

Visual Basic [form.]control.EnableMouseEvents[= boolean%]

Designer Page

[Miscellaneous designer page](#)

Remarks

The default value for the EnableMouseEvents property is True, which triggers the standard Visual Basic MouseDown, MouseMove, and MouseUp events. When the EnableMouseEvents property is set to False, the mouse events do not occur.

Disabling these events speeds up the performance of the fpCombo and fpList controls.

Data Type

Integer (Boolean)

See Also

[EnableKeyEvents](#), [EnableTopChangeEvent](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

EnableTopChangeEvent Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether the [TopChange](#) event occurs for an fpCombo or fpList control. This property is available for ActiveX and VBX controls only.

Syntax

C++ **BOOL Class::GetEnableTopChangeEvent(void);**
 Class::SetEnableTopChangeEvent(BOOL value);

Visual Basic [form.]control.EnableTopChangeEvent[= boolean%]

Designer Page

[Miscellaneous designer page](#)

Remarks

The default value for the EnableTopChangeEvent property is True, which triggers the TopChange event. When the EnableTopChangeEvent property is set to False, the TopChange event does not occur.

Disabling the TopChange event speeds up the performance of the fpCombo and fpList controls.

Data Type

Integer (Boolean)

See Also

[EnableKeyEvents](#), [EnableMouseEvents](#) properties

[TopChange](#) event

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ExtendCol Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether column borders are displayed beyond the last row in a multiple-column control.

Syntax

C *UINT* LC_GetExtendCol(*HWND* hWnd, *short FAR* *pValue);
 UINT LC_SetExtendCol(*HWND* hWnd, *short* value);

C++ *short* **Class**::GetExtendCol(*void*);
 Class::SetExtendCol(*short* value);

Visual Basic [form.]control.ExtendCol[= setting%]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Auto	(Default) If horizontal lines are displayed, columns extend to the last row; otherwise, columns extend to the control border	LC_EXTENDCOL_AUTO
1 - No	Columns extend to the last row	LC_EXTENDCOL_NO
2 - Yes	Columns extend to the control border	LC_EXTENDCOL_YES

Use the [ExtendRow](#) property in the same manner to extend rows and row selection.

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a list box control with three columns. Flat lines are displayed between the columns and both columns and rows are extended.

C

```
LC_SetColumns(hWnd, 3);
LC_SetLineApplyTo(hWnd, LC_LINEAPPLYTO_COLS);
LC_SetLineStyle(hWnd, LC_LINESTYLE_FLAT);
LC_SetExtendCol(hWnd, LC_EXTENDCOL_YES);
LC_SetExtendRow(hWnd, LC_EXTENDROW_YES);
```

C++

```
fpList1->SetColumns(3);
fpList1->SetLineApplyTo(LC_LINEAPPLYTO_COLS);
fpList1->SetLineStyle(LC_LINESTYLE_FLAT);
fpList1->SetExtendCol(LC_EXTENDCOL_YES);
fpList1->SetExtendRow(LC_EXTENDROW_YES);
```

Visual Basic

```
fpList1.Columns = 3
fpList1.LineApplyTo = LC_LINEAPPLYTO_COLS
fpList1.LineStyle = LC_LINESTYLE_FLAT
fpList1.ExtendCol = LC_EXTENDCOL_YES
fpList1.ExtendRow = LC_EXTENDROW_YES
```

See Also

[Customizing Lines](#)

[ExtendRow](#), [LineApplyTo](#), [LineStyle](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ExtendRow Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether rows and row selection are displayed beyond the last column in multiple-column controls.

Syntax

C *UINT* LC_GetExtendRow(*HWND* hWnd, *short FAR* *lpValue);
 UINT LC_SetExtendRow(*HWND* hWnd, *short* value);

C++ *short Class::*GetExtendRow(*void*);
 *Class::*SetExtendRow(*short* value);

Visual Basic [form.]control.ExtendRow[= setting%]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Auto	(Default) If vertical lines are displayed, rows and row selection extend to the last column; otherwise, rows and row selection extend to the control border	LC_EXTENDROW_AUTO
1 - No	Rows and row selection extend to the last column	LC_EXTENDROW_NO
2 - Yes	Rows and row selection extend to the control border	LC_EXTENDROW_YES

Use the [ExtendCol](#) property in the same manner to extend columns.

Data Type

Integer (Enumerated)

See Also

[Customizing Lines](#)

[ExtendCol](#), [LineApplyTo](#), [LineStyle](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Font Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns font characteristics of displayed text. This property is available for ActiveX controls only.

Designer Page

[List subtab of the ApplyTo designer page](#)

ActiveX Use

The Font property is a stock property.

Remarks

The List Pro ActiveX controls combine the FontBold, FontItalic, FontName, FontSize, FontStrikethru, and FontUnderline properties into the Font property. However, the ActiveX controls still support these properties in existing projects.

The attributes of the Font property default to the following values:

Attribute	Default value
Bold	False
Italic	False
Name	MS Sans Serif
Size	10
Strikethru	False
Underline	False

If you double-click the Font property in some property browsers, the control displays the Fonts dialog box for setting font characteristics. You can also set font characteristics in code.

Data Type

Font

[Print](#)

[Copy](#)

[Close](#)

The following example sets the font to be bold in an fpCombo control.

ActiveX

```
fpCombo1.Font.Bold = True
```

See Also

[Changing Text Color and Fonts](#)

[FontEmpty](#) property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

FontEmpty Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether the current font attributes for a part of the control are cleared and the new font attributes are inherited from the part's hierarchical predecessor.

Syntax

C *UINT* LC_GetFontEmpty(*HWND* hWnd, *BOOL FAR* *pValue);
UINT LC_SetFontEmpty(*HWND* hWnd, *BOOL* value);

C++ *BOOL* *Class*::GetFontEmpty(*void*);
Class::SetFontEmpty(*BOOL* value);

Visual Basic [form.]control.FontEmpty[= *boolean%*]

Designer Page

[List subtab of the ApplyTo designer page](#) (Clear Font button in property value area for the Font property)

Remarks

The default value for the FontEmpty property is False. When the FontEmpty property is set to True for a part of the control, the font attributes for that part are cleared and the font attributes are inherited from the part's hierarchical predecessor. Setting the FontEmpty property to True is like the Default setting for other designated-list properties such as [AlignH](#) and [TextOrientation](#).

For more information about the hierarchy of property settings, refer to Appendix C "Hierarchy of Property Settings" of the printed *List Pro User's Guide*.

You can use the [ListApplyTo](#) property to specify where the FontEmpty property applies.

Data Type

Integer (Boolean)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a combo box control that has three columns. The first column is a child of the first group. The first column inherits the font characteristics of its parent group.

C

```
LC_SetColumns(hWnd, 3);
LC_SetColumnHeaderShow(hWnd, FALSE);
LC_SetGroups(hWnd, 2);
LC_SetGroupHeaderShow(hWnd, TRUE);
LC_SetGrp(hWnd, 0);
LC_SetGrpHeaderText(hWnd, "Title");
LC_SetGrp(hWnd, 1);
LC_SetGrpHeaderText(hWnd, "ISBN");
/* Assign columns to groups */
LC_SetCol(hWnd, 0);
LC_SetColParentGroup(hWnd, 0);
LC_SetCol(hWnd, 1);
LC_SetColParentGroup(hWnd, 1);
LC_SetCol(hWnd, 2);
LC_SetColParentGroup(hWnd, 1);
/* Make font for group 0 italic */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SINGLE_GROUP);
LC_SetGrp(hWnd, 0);
LC_SetFontItalic(hWnd, TRUE);
/* Clear fonts from column 0 (inherit from parent group) */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SINGLE_ITEM);
LC_SetCol(hWnd, 0);
LC_SetFontEmpty(hWnd, TRUE);
```

C++

```
fpCombo1->SetColumns(3);
fpCombo1->SetColumnHeaderShow(FALSE);
fpCombo1->SetGroups(2);
fpCombo1->SetGroupHeaderShow(TRUE);
fpCombo1->SetGrp(0);
fpCombo1->SetGrpHeaderText("Title");
fpCombo1->SetGrp(1);
fpCombo1->SetGrpHeaderText("ISBN");
// Assign columns to groups
fpCombo1->SetCol(0);
fpCombo1->SetColParentGroup(0);
fpCombo1->SetCol(1);
fpCombo1->SetColParentGroup(1);
fpCombo1->SetCol(2);
fpCombo1->SetColParentGroup(1);
// Make font for group 0 italic
fpCombo1->SetListApplyTo(LC_LISTAPPLYTO_SINGLE_GROUP);
fpCombo1->SetGrp(0);
fpCombo1->SetFontItalic(TRUE);
// Clear fonts from column 0 (inherit from parent group)
fpCombo1->SetListApplyTo(LC_LISTAPPLYTO_SINGLE_ITEM);
fpCombo1->SetCol(0);
fpCombo1->SetFontEmpty(TRUE);
```

Visual Basic

```
fpCombo1.Columns = 3
fpCombo1.ColumnHeaderShow = False
fpCombo1.Groups = 2
fpCombo1.GroupHeaderShow = True
fpCombo1.Grp = 0
fpCombo1.GrpHeaderText = "Title"
fpCombo1.Grp = 1
fpCombo1.GrpHeaderText = "ISBN"
```

```
' Assign columns to groups
fpCombol.Col = 0
fpCombol.ColParentGroup = 0
fpCombol.Col = 1
fpCombol.ColParentGroup = 1
fpCombol.Col = 2
fpCombol.ColParentGroup = 1
' Make font for group 0 italic
fpCombol.ListApplyTo = LC_LISTAPPLYTO_SINGLE_GROUP
fpCombol.Grp = 0
fpCombol.FontItalic = True
' Clear fonts from column 0 (inherit from parent group)
fpCombol.ListApplyTo = LC_LISTAPPLYTO_SINGLE_ITEM
fpCombol.Col = 0
fpCombol.FontEmpty = True
```

See Also

[Changing Text Color and Fonts](#)

[Font](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

GrayAreaColor Property

[See Also](#)

[Example](#)

Applies To

fpCombo control

Description

Sets or returns the color of the gray area surrounding drop-down combo and simple combo style fpCombo controls.

Syntax

C **UINT** CbxGetGrayAreaColor(*HWND* hWnd, **COLORREF FAR** *lpValue);
UINT CbxSetGrayAreaColor(*HWND* hWnd, **COLORREF** value);

C++ **COLORREF** CfpComboBox::GetGrayAreaColor(**void**);
CfpComboBox::SetGrayAreaColor(**COLORREF** value);

Visual Basic [*form.*]fpCombo1.GrayAreaColor[= color&]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

The default value for the GrayAreaColor property is &H8000000F&.

This property has no effect if the [Style](#) property is set to 2 (Drop-Down List).

Data Type

Color

See Also

[Choosing the fpCombo Control Style](#)

[Style](#) property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

GroupHeaderHeight Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the height of the group header text.

Syntax

C `UINT LC_GetGroupHeaderHeight(HWND hWnd, long FAR *lpValue);`
`UINT LC_SetGroupHeaderHeight(HWND hWnd, long value);`

C++ `long Class::GetGroupHeaderHeight(void);`
`Class::SetGroupHeaderHeight(long value);`

Visual Basic `[form.]control.GroupHeaderHeight[= value!]`

Designer Page

[General subtab of the Groups designer page](#)

Remarks

The default value for the GroupHeaderHeight property is -1, which specifies that the group header height is automatically sized to fit the text.

In Visual Basic, the measurement unit used by the GroupHeaderHeight property depends on the setting of the form's ScaleMode property. The default ScaleMode setting is twips (1/1440 of an inch). Generally the ActiveX and VBX controls use twips as the default measurement unit, and the DLL control uses pixels as the default measurement unit.

Data Type

Single

[Print](#)

[Copy](#)

[Close](#)

The following example creates a combo box control with two groups. The group header height of the first group is 600 twips. Group header text is displayed on multiple lines.

C

```
LC_SetGroups(hWnd, 2);
LC_SetGroupHeaderShow(hWnd, TRUE);
/* Set group header height */
LC_SetGroupHeaderHeight(hWnd, 60);
/* Display text on multiple lines */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_GROUP_HEADERS);
LC_SetMultiLine(hWnd, LC_MULTILINE_MULTIPLE_LINE);
LC_SetGrp(hWnd, 0);
LC_SetGrpHeaderText(hWnd, "Date Service Received From Provider");
LC_SetGrp(hWnd, 1);
LC_SetGrpHeaderText(hWnd, "Service Provider");
```

C++

```
fpCombo1->SetGroups(2);
fpCombo1->SetGroupHeaderShow(TRUE);
// Set group header height
fpCombo1->SetGroupHeaderHeight(60);
// Display text on multiple lines
fpCombo1->SetListApplyTo(LC_LISTAPPLYTO_GROUP_HEADERS);
fpCombo1->SetMultiLine(LC_MULTILINE_MULTIPLE_LINE);
fpCombo1->SetGrp(0);
fpCombo1->SetGrpHeaderText("Date Service Received From Provider");
fpCombo1->SetGrp(1);
fpCombo1->SetGrpHeaderText("Service Provider");
```

Visual Basic

```
fpCombo1.Groups = 2
fpCombo1.GroupHeaderShow = True
' Set group header height
fpCombo1.GroupHeaderHeight = 600
' Display text on multiple lines
fpCombo1.ListApplyTo = LC_LISTAPPLYTO_GROUP_HEADERS
fpCombo1.MultiLine = LC_MULTILINE_MULTIPLE_LINE
fpCombo1.Grp = 0
fpCombo1.GrpHeaderText = "Date Service Received From Provider"
fpCombo1.Grp = 1
fpCombo1.GrpHeaderText = "Service Provider"
```

See Also

[Creating Group Headers](#)

[GroupHeaderShow](#), [Groups](#), [Grp](#), [GrpHeaderText](#), [GrpHide](#), [MultiLine](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

GroupHeaderShow Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether to display the header text for a group.

Syntax

C `UINT LC_GetGroupHeaderShow(HWND hWnd, BOOL FAR *pValue);`
`UINT LC_SetGroupHeaderShow(HWND hWnd, BOOL value);`

C++ `BOOL Class::GetGroupHeaderShow(void);`
`Class::SetGroupHeaderShow(BOOL value);`

Visual Basic `[form.]control.GroupHeaderShow[= boolean%]`

Designer Page

[General subtab of the Groups designer page](#)

Remarks

The default value for the GroupHeaderShow property is False. To show the group header text, set the GroupHeaderShow property to True.

Use the [GrpHeaderText](#) property to specify the header text.

Note You must display group headers to drag and drop groups.

Data Type

Integer (Boolean)

See Also

[Creating Group Headers](#)

[GroupHeaderHeight](#), [Groups](#), [Grp](#), [GrpHeaderText](#), [GrpHide](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Groups Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the number of groups in the control.

Syntax

C	UINT LC_GetGroups(HWND hWnd, short FAR *lpValue); UINT LC_SetGroups(HWND hWnd, short value);
C++	short Class:: GetGroups(void); Class:: SetGroups(short value);
Visual Basic	[form.]control.Groups[= value%]

Designer Page

[Specific subtab of the Groups designer page](#)

Remarks

The default value for the Groups property is 0.

Columns in multiple-column fpCombo and fpList controls can be grouped together. These columns are *children* of the group. A child of a group can be another group ([GrpParentGroup](#) property) or a column ([ColParentGroup](#) property), but not both at the same time. A column can be a child of a group that is a child of another group. For more information about groups and how they work, see [Groups](#).

Children of groups exhibit the following characteristics:

- If you move a group, the group's children move with it.
- When you hide a group ([GrpHide](#) property), the group's children are also hidden.
- Children are automatically sized to fit the group width ([GrpWidth](#) property). This can result in text not being fully displayed in a column or group header.

Tip If you are using groups, all columns should be assigned to a group.

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example creates a list box control with four groups. Group 3 is hidden and Group 1 is locked against resizing.

C

```
LC_SetGroups(hWnd, 4);
LC_SetGroupHeaderShow(hWnd, TRUE);
LC_SetAllowGrpResize(hWnd, LC_ALLOWGRPRESIZE_RESIZEHEADER);
/* Define groups */
LC_SetGrp(hWnd, 0);
LC_SetGrpID(hWnd, 1);
LC_SetGrp(hWnd, 1);
LC_SetGrpID(hWnd, 2);
LC_SetGrp(hWnd, 2);
LC_SetGrpID(hWnd, 3);
LC_SetGrp(hWnd, 3);
LC_SetGrpID(hWnd, 4);
/* Hide group 3 */
LC_SetGrpFromID(hWnd, 3);
LC_SetGrpHide(hWnd, TRUE);
/* Lock group 1 against resizing */
LC_SetGrpFromID(hWnd, 1);
LC_SetGrpLockResize(hWnd, TRUE);
```

C++

```
fpList1->SetGroups(4);
fpList1->SetGroupHeaderShow(TRUE);
fpList1->SetAllowGrpResize(LC_ALLOWGRPRESIZE_RESIZEHEADER);
// Define groups
fpList1->SetGrp(0);
fpList1->SetGrpID(1);
fpList1->SetGrp(1);
fpList1->SetGrpID(2);
fpList1->SetGrp(2);
fpList1->SetGrpID(3);
fpList1->SetGrp(3);
fpList1->SetGrpID(4);
// Hide group 3
fpList1->SetGrpFromID(3);
fpList1->SetGrpHide(TRUE);
// Lock group 1 against resizing
fpList1->SetGrpFromID(1);
fpList1->SetGrpLockResize(TRUE);
```

Visual Basic

```
fpList1.Groups = 4
fpList1.GroupHeaderShow = True
fpList1.AllowGrpResize = LC_ALLOWGRPRESIZE_RESIZEHEADER
' Define groups
fpList1.Grp = 0
fpList1.GrpID = 1
fpList1.Grp = 1
fpList1.GrpID = 2
fpList1.Grp = 2
fpList1.GrpID = 3
fpList1.Grp = 3
fpList1.GrpID = 4
' Hide group 3
fpList1.GrpFromID = 3
fpList1.GrpHide = True
' Lock group 1 against resizing
fpList1.GrpFromID = 1
fpList1.GrpLockResize = True
```

See Also

[Creating Groups](#)

[Groups](#)

[ColParentGroup](#), [GroupHeaderHeight](#), [GroupHeaderShow](#), [Grp](#), [GrpFromID](#), [GrpFromName](#), [GrpHeaderText](#), [GrpHide](#), [GrpID](#), [GrpLockResize](#), [GrpName](#), [GrpParentGroup](#), [GrpPos](#), [GrpPosInParent](#), [GrpsFrozen](#), [GrpWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	

Grp Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the index number of a group in an fpCombo or fpList control.

Syntax

C	UINT LC_GetGrp(HWND hWnd, short FAR *lpValue); UINT LC_SetGrp(HWND hWnd, short value);
C++	short Class ::GetGrp(void); Class ::SetGrp(short value);
Visual Basic	[form.]control.Grp[= value%]

Designer Pages

Grp drop-down list box on:

[Specific subtab of the Groups designer page](#)

[List subtab of the ApplyTo designer page](#)

[Line subtab of the ApplyTo designer page](#)

Remarks

The default value for the Grp property is 0.

The Grp property specifies the group on which the designated-group properties (such as [GrpHeaderText](#) and [GrpHide](#)) operate. Group numbers start with 0, which designates the top, leftmost group. Groups are numbered from left to right and top to bottom within their parent group, if any, and then within the control.

Group index numbers are based on the physical position of the group in the control. For example, assume you define three groups (0, 1, and 2). These groups appear in that order from left to right across the top of the control. If you move Group 2 to the far left side of the control, this group now has a group index number of 0. For more information on groups, see [Referencing a Group](#).

You must set the [Groups](#) property to a value greater than zero before using this property.

You can use the [GrpID](#) and [GrpName](#) properties to assign unique identifiers to a group. You can then use the [GrpFromID](#) or [GrpFromName](#) properties to specify the group to which the designated-group properties apply, regardless of where the group physically appears in the control.

Tip Because group index numbers are based on the physical position of the group in the control, if you plan on moving groups or allowing the user to move groups, we strongly recommend you use one of the unique group identifiers (GrpID or GrpName property) to reference a group rather than the Grp property.

Data Type

Integer

See Also

[Referencing a Group](#)

[Using the FarPoint Property Designer](#)

[GroupHeaderHeight](#), [GroupHeaderShow](#), [Groups](#), [GrpFromID](#), [GrpFromName](#), [GrpHeaderText](#), [GrpHide](#), [GrpID](#), [GrpLockResize](#), [GrpName](#), [GrpParentGroup](#), [GrpPos](#), [GrpPosInParent](#), [GrpWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	

GrpFromID Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the group on which the designated-group properties operate by using the group identifier number.

Syntax

C	UINT LC_GetGrpFromID(HWND hWnd, long FAR *lpValue); UINT LC_SetGrpFromID(HWND hWnd, long value);
C++	long Class ::GetGrpFromID(void); Class ::SetGrpFromID(long value);
Visual Basic	[form.]control.GrpFromID[= value&]

Designer Pages

Grp ID drop-down list box on:

[Specific subtab of the Groups designer page](#)

[List subtab of the ApplyTo designer page](#)

[Line subtab of the ApplyTo designer page](#)

Remarks

The default value for the GrpFromID property is 0.

You must set the [GrpID](#) property to create the group identifier number before setting the GrpFromID property. Once you define an identifier number for a group, you can use the GrpFromID property to specify the group on which the designated-group properties (such as [GrpHeaderText](#) and [GrpHide](#)) operate. The GrpFromID property works the same as the Grp property in this respect.

You can also use the [GrpName](#) and [GrpFromName](#) properties in a similar fashion.

Tip Because group index numbers are based on the physical position of the group in the control, if you plan on moving groups, we strongly recommend you use one of the unique group identifiers (GrpID or GrpName property) to reference a group rather than the [Grp](#) property.

Data Type

Integer (Long)

See Also

[Applying Properties to a Specific Group](#)

[Referencing a Group](#)

[Grp](#), [GrpFromName](#), [GrpID](#), [GrpName](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	

GrpFromName Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the group on which the designated-group properties operate by using the group name.

Syntax

C *UINT* LC_GetGrpFromName(*HWND* hWnd, *LPSTR* buffer, *UINT* nBufferSize);
 UINT LC_SetGrpFromName(*HWND* hWnd, *LPCSTR* value);

C++ *LPSTR Class::*GetGrpFromName(*LPSTR* buffer, *UINT* bufferSize);
 *Class::*SetGrpFromName(*LPCSTR* value);

Visual Basic [form.]control.GrpFromName[= text\$]

Designer Pages

Grp Name drop-down list box on:

[Specific subtab of the Groups designer page](#)
[List subtab of the ApplyTo designer page](#)
[Line subtab of the ApplyTo designer page](#)

Remarks

You must set the [GrpName](#) property to define the group name before setting the GrpFromName property. Once you define a name for a group, you can use the GrpFromName property to specify the group on which the designated-group properties (such as [GrpHeaderText](#) and [GrpHide](#)) operate. The GrpFromName property works the same as the [Grp](#) property in this respect.

The GrpName property is case-sensitive; however, the GrpFromName property is not case-sensitive.

You can also use the [GrpID](#) and [GrpFromID](#) properties in a similar fashion.

Tip Because group index numbers are based on the physical position of the group in the control, if you plan on moving groups, we strongly recommend you use one of the unique group identifiers (GrpID or GrpName property) to reference a group rather than the Grp property.

Data Type

String

[Print](#)

[Copy](#)

[Close](#)

The following example creates a list box control with five groups. Groups 3 and 4 are children of the first group. Group 5 is a child of the second group.

C

```
LC_SetGroups(hWnd, 5);
LC_SetGroupHeaderShow(hWnd, TRUE);
LC_SetLineStyle(hWnd, LC_LINESTYLE_LOWERED);
LC_SetGroupHeaderShow(hWnd, TRUE);
/* Define parent groups */
LC_SetGrp(hWnd, 0);
LC_SetGrpHeaderText(hWnd, "Management");
LC_SetGrp(hWnd, 1);
LC_SetGrpHeaderText(hWnd, "Human Resources");
/* Define child groups */
LC_SetGrp(hWnd, 2);
/* Name */
LC_SetGrpName(hWnd, "A1");
LC_SetGrp(hWnd, 3);
/* Title */
LC_SetGrpName(hWnd, "A2");
LC_SetGrp(hWnd, 4);
/* Salary */
LC_SetGrpName(hWnd, "A3");
/* Define position in control */
LC_SetGrpFromName(hWnd, "A3");
LC_SetGrpHeaderText(hWnd, "Salary");
LC_SetGrpPos(hWnd, 2);
LC_SetGrpFromName(hWnd, "A2");
LC_SetGrpHeaderText(hWnd, "Title");
LC_SetGrpPos(hWnd, 3);
LC_SetGrpFromName(hWnd, "A1");
LC_SetGrpHeaderText(hWnd, "Name");
LC_SetGrpPos(hWnd, 4);
/* Define parent group and position in parent */
/* Title column */
LC_SetGrpFromName(hWnd, "A2");
LC_SetGrpParentGroup(hWnd, 0);
LC_SetGrpPosInParent(hWnd, 0);
/* Name column */
LC_SetGrpFromName(hWnd, "A1");
LC_SetGrpParentGroup(hWnd, 0);
LC_SetGrpPosInParent(hWnd, 1);
/* Salary column */
LC_SetGrpFromName(hWnd, "A3");
LC_SetGrpParentGroup(3);
LC_SetGrpPosInParent(hWnd, 0);
```

C++

```
fpList1->SetGroups(5);
fpList1->SetGroupHeaderShow(TRUE);
fpList1->SetLineStyle(LC_LINESTYLE_LOWERED);
fpList1->SetGroupHeaderShow(TRUE);
// Define parent groups
fpList1->SetGrp(0);
fpList1->SetGrpHeaderText("Management");
fpList1->SetGrp(1);
fpList1->SetGrpHeaderText("Human Resources");
// Define child groups
fpList1->SetGrp(2);
// Name
fpList1->SetGrpName("A1");
fpList1->SetGrp(3);
```



```

// Title
fpList1->SetGrpName("A2");
fpList1->SetGrp(4);
// Salary
fpList1->SetGrpName("A3");
// Define position in control
fpList1->SetGrpFromName("A3");
fpList1->SetGrpHeaderText("Salary");
fpList1->SetGrpPos(2);
fpList1->SetGrpFromName("A2");
fpList1->SetGrpHeaderText("Title");
fpList1->SetGrpPos(3);
fpList1->SetGrpFromName("A1");
fpList1->SetGrpHeaderText("Name");
fpList1->SetGrpPos(4);
// Define parent group and position in parent
// Title column
fpList1->SetGrpFromName("A2");
fpList1->SetGrpParentGroup(0);
fpList1->SetGrpPosInParent(0);
// Name column
fpList1->SetGrpFromName("A1");
fpList1->SetGrpParentGroup(0);
fpList1->SetGrpPosInParent(1);
// Salary column
fpList1->SetGrpFromName("A3");
fpList1->SetGrpParentGroup(3);
fpList1->SetGrpPosInParent(0);

```

Visual Basic

```

fpList1.Groups = 5
fpList1.GroupHeaderShow = True
fpList1.LineStyle = LC_LINESTYLE_LOWERED
fpList1.GroupHeaderShow = True
' Define parent groups
fpList1.Grp = 0
fpList1.GrpHeaderText = "Management"
fpList1.Grp = 1
fpList1.GrpHeaderText = "Human Resources"
' Define child groups
fpList1.Grp = 2
' Name
fpList1.GrpName = "A1"
fpList1.Grp = 3
' Title
fpList1.GrpName = "A2"
fpList1.Grp = 4
' Salary
fpList1.GrpName = "A3"
' Define position in control
fpList1.GrpFromName = "A3"
fpList1.GrpHeaderText = "Salary"
fpList1.GrpPos = 2
fpList1.GrpFromName = "A2"
fpList1.GrpHeaderText = "Title"
fpList1.GrpPos = 3
fpList1.GrpFromName = "A1"
fpList1.GrpHeaderText = "Name"
fpList1.GrpPos = 4
' Define parent group and position in parent
' Title column
fpList1.GrpFromName = "A2"
fpList1.GrpParentGroup = 0

```

```
fpList1.GrpPosInParent = 0
' Name column
fpList1.GrpFromName = "A1"
fpList1.GrpParentGroup = 0
fpList1.GrpPosInParent = 1
' Salary column
fpList1.GrpFromName = "A3"
fpList1.GrpParentGroup= 3
fpList1.GrpPosInParent = 0
```

See Also

[Applying Properties to a Specific Group](#)

[Referencing a Group](#)

[Grp](#), [GrpFromID](#), [GrpID](#), [GrpName](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	

GrpHeaderText Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the string to display in the group header.

Syntax

C **UINT** LC_GetGrpHeaderText(**HWND** hWnd, **LPSTR** buffer, **UINT** nBufferSize);
UINT LC_SetGrpHeaderText(**HWND** hWnd, **LPCSTR** value);

C++ **LPSTR** **Class**::GetGrpHeaderText(**LPSTR** buffer, **UINT** bufferSize);
Class::SetGrpHeaderText(**LPCSTR** value);

Visual Basic [form.]control.GrpHeaderText[= text\$]

Designer Page

[Specific subtab of the Groups designer page](#)

Remarks

The default value for the GrpHeaderText property is an empty string.

You must set the [GroupHeaderShow](#) property to True to display the group headers.

Before you set the GrpHeaderText property, you must specify a group with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Data Type

String

See Also

[Creating Group Headers](#)

[GroupHeaderHeight](#), [GroupHeaderShow](#), [Groups](#), [Grp](#), [GrpFromID](#), [GrpFromName](#), [GrpHide](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	

GrpHide Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether to display a specific group and its associated children.

Syntax

C *UINT* LC_GetGrpHide(*HWND* hWnd, *BOOL FAR* *lpValue);
 UINT LC_SetGrpHide(*HWND* hWnd, *BOOL* value);

C++ *BOOL* *Class*::GetGrpHide(*void*);
 Class::SetGrpHide(*BOOL* value);

Visual Basic [form.]control.GrpHide[= *boolean%*]

Designer Page

[Specific subtab of the Groups designer page](#)

Remarks

The default value for the GrpHide property is False. To hide a group, set the GrpHide property to True.

Before you set the GrpHide property, you must specify a group with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Tip Another way to hide a group is to set the [GrpWidth](#) property to 0.

Data Type

Integer (Boolean)

See Also

[Customizing Groups](#)

[GroupHeaderShow](#), [Groups](#), [Grp](#), [GrpFromID](#), [GrpFromName](#), [GrpHeaderText](#), [GrpID](#), [GrpName](#), [GrpParentGroup](#), [GrpWidth](#)
properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	

GrpID Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the group identifier number.

Syntax

C **UINT** LC_GetGrpID(**HWND** hWnd, **long FAR** *lpValue);
UINT LC_SetGrpID(**HWND** hWnd, **long** value);

C++ **long** **Class**::GetGrpID(**void**);
Class::SetGrpID(**long** value);

Visual Basic [form.]control.GrpID[= value&]

Designer Page

[Specific subtab of the Groups designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the GrpID property is 1.

Before you set the GrpID property, you must specify a group with the [Grp](#) property.

The GrpID property defines a unique identifier number for a group. Once you define an identifier number for a group, you can use the [GrpFromID](#) property to specify the group on which the designated-group properties (such as [GrpHeaderText](#) and [GrpHide](#)) operate. The GrpFromID property works the same as the Grp property in this respect.

You can also use the [GrpName](#) and [GrpFromName](#) properties in a similar fashion.

Tip Because group index numbers are based on the physical position of the group in the control, if you plan on moving groups, we strongly recommend you use one of the unique group identifiers (GrpID or GrpName property) to reference a group rather than the Grp property.

Data Type

Integer (Long)

See Also

[Applying Properties to a Specific Group](#)

[Referencing a Group](#)

[Grp](#), [GrpFromID](#), [GrpFromName](#), [GrpName](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	

GrpLockResize Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether a group is locked and cannot be resized by dragging the group border.

Syntax

C `UINT LC_GetGrpLockResize(HWND hWnd, BOOL FAR *lpValue);`
UINT LC_SetGrpLockResize(**HWND** hWnd, **BOOL** value);

C++ `BOOL Class::GetGrpLockResize(void);`
Class::SetGrpLockResize(**BOOL** value);

Visual Basic `[form.]control.GrpLockResize[= boolean%]`

Designer Page

[Specific subtab of the Groups designer page](#)

Remarks

When the [AllowGrpResize](#) property is set to a value greater than zero, the user can resize groups by dragging their borders. When set to True, the GrpLockResize property locks a specific group, preventing it from being resized.

The default value for the GrpLockResize property is False. Set the GrpLockResize property to True to prevent the user from dragging the right border of a group to resize it.

Before you set the GrpLockResize property, you must specify a group with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Data Type

Integer (Boolean)

See Also

[Customizing Groups](#)

[Using the Mouse to Resize Groups](#)

[AllowGrpResize](#), [Groups](#), [Grp](#), [GrpFromID](#), [GrpFromName](#), [GrpID](#), [GrpName](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

GrpName Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the group name.

Syntax

C	<i>UINT</i> LC_GetGrpName(<i>HWND</i> hWnd, <i>LPSTR</i> buffer, <i>UINT</i> nBufferSize); <i>UINT</i> LC_SetGrpName(<i>HWND</i> hWnd, <i>LPCSTR</i> value);
C++	<i>LPSTR Class::</i> GetGrpName(<i>LPSTR</i> buffer, <i>UINT</i> bufferSize); <i>Class::</i> SetGrpName(<i>LPCSTR</i> value);
Visual Basic	[form.]control.GrpName[= text\$]

Designer Page

[Specific subtab of the Groups designer page](#)

Remarks

You must set the [Grp](#) property to specify a group before setting the GrpName property.

The GrpName property defines a unique name for a group. Once you define a name for a group, you can use the [GrpFromName](#) property to specify the group on which the designated-group properties (such as [GrpHeaderText](#) and [GrpHide](#)) operate. The GrpFromName property works the same as the Grp property in this respect.

The GrpName property is case-sensitive; however, the GrpFromName property is not case-sensitive.

You can also use the [GrpID](#) and [GrpFromID](#) properties in a similar fashion.

Tip Because group index numbers are based on the physical position of the group in the control, if you plan on moving groups, we strongly recommend you use one of the unique group identifiers (GrpID or GrpName property) to reference a group rather than the Grp property.

Data Type

String

See Also

[Applying Properties to a Specific Group](#)

[Referencing a Group](#)

[Grp](#), [GrpFromID](#), [GrpFromName](#), [GrpID](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

GrpParentGroup Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the number of the parent group of another group.

Syntax

C **UINT** LC_GetGrpParentGroup(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetGrpParentGroup(**HWND** hWnd, **short** value);

C++ **short Class::**GetGrpParentGroup(**void**);
Class::SetGrpParentGroup(**short** value);

Visual Basic [form.]control.GrpParentGroup[= value%]

Designer Page

[Specific subtab of the Groups designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the GrpParentGroup property is 1, which specifies that a group has no parent. Before you set the GrpParentGroup property, you must specify a group with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property. Groups can have children. A child of a group can be another group or a column ([ColParentGroup](#) property), but not both at the same time. A column can be a child of a group that is a child of another group. For more information about groups and how they work, see [Groups](#).

Children of groups exhibit the following characteristics:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If you move a group, the group's children move with it.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

When you hide a group ([GrpHide](#) property), the group's children are also hidden.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Children are automatically sized to fit the group width ([GrpWidth](#) property). This can result in text not being fully displayed in a column or group header.

Use the GrpParentGroup property to specify a group as a child of another group.

Tips

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

GrpParentGroup property.

Define all groups (Grp, [GrpID](#), and [GrpName](#) properties) in the control before moving groups with the

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Because group index numbers are based on the physical position of the group in the control, if you plan on moving groups, we strongly recommend you use one of the unique group identifiers (GrpID or GrpName property) to reference a group rather than the Grp property.

Data Type

Integer

See Also

[Creating Children of Groups](#)

[Groups](#)

[ColParentGroup](#), [GroupHeaderShow](#), [Groups](#), [Grp](#), [GrpFromID](#), [GrpFromName](#), [GrpHide](#), [GrpID](#), [GrpLockResize](#), [GrpName](#), [GrpPosInParent](#), [GrpWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

GrpPos Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the position number of a group within an fpCombo or fpList control.

Syntax

C **UINT** LC_GetGrpPos(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetGrpPos(**HWND** hWnd, **short** value);

C++ **short Class::**GetGrpPos(**void**);
Class::SetGrpPos(**short** value);

Visual Basic [form.]control.GrpPos[= value%]

Designer Page

[Specific subtab of the Groups designer page](#)

Remarks

The default value for the GrpPos property is 0.

Before you set the GrpPos property, you must specify a group with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Use the [GrpPosInParent](#) property to define the position number of a group within its parent. If groups are grouped, the GrpPosInParent property defines the position number within the group's parent group. If groups are not grouped, the value of the GrpPosInParent property is the same as the value of the GrpPos property.

Tips

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

GrpParentGroup property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Define all groups (Grp, [GrpID](#), and [GrpName](#) properties) in the control before moving groups with the

GrpParentGroup property. Because group index numbers are based on the physical position of the group in the control, if you plan on moving groups, we strongly recommend you use one of the unique group identifiers (GrpID or GrpName property) to reference a group rather than the Grp property.

Note If you move a group, group index numbers will change. For more information, see [Referencing a Group](#).

Data Type

Integer

See Also

[Defining the Position of Groups in the Control](#)

[Referencing a Group](#)

[Groups](#), [Grp](#), [GrpFromID](#), [GrpFromName](#), [GrpID](#), [GrpName](#), [GrpParentGroup](#), [GrpPosInParent](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

GrpPosInParent Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the position number of a group within its parent.

Syntax

C *UINT* LC_GetGrpPosInParent(*HWND* hWnd, *short FAR* *lpValue);
 UINT LC_SetGrpPosInParent(*HWND* hWnd, *short* value);

C++ *short Class::*GetGrpPosInParent(*void*);
 *Class::*SetGrpPosInParent(*short* value);

Visual Basic [form.]control.GrpPosInParent[= value%]

Designer Page

[Specific subtab of the Groups designer page](#)

Remarks

The default value for the GrpPosInParent property is 0 when the [Groups](#) property is set to a value greater than 0. Otherwise, the

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

default value is 1.

Before you set the GrpPosInParent property, you must specify a column with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property. If groups are grouped, the GrpPosInParent property defines the position number within the group's parent group. Use the [GrpPos](#) property to define the position number of a group within the control. If groups are not grouped, the value of the GrpPosInParent property is the same as the value of the GrpPos property.

Tips

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Define all groups (Grp, [GrpID](#), and [GrpName](#) properties) in the control before moving groups with the GrpParentGroup property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Because group index numbers are based on the physical position of the group in the control, if you plan on moving groups, we strongly recommend you use one of the unique group identifiers (GrpID or GrpName property) to reference a group rather than the Grp property.

Note If you move a group, group index numbers will change. For more information, see [Referencing a Group](#).

Data Type

Integer

See Also

[Defining the Position of Groups in the Control](#)

[Referencing a Group](#)

[Groups](#), [Grp](#), [GrpFromID](#), [GrpFromName](#), [GrpID](#), [GrpName](#), [GrpParentGroup](#), [GrpPos](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

GrpsFrozen Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the number of groups on the left that do not scroll horizontally in a multiple-group fpCombo or fpList control.

Syntax

C *UINT* LC_GetGrpsFrozen(*HWND* hWnd, *short FAR* *lpValue);
 UINT LC_SetGrpsFrozen(*HWND* hWnd, *short* value);

C++ *short Class::*GetGrpsFrozen(*void*);
 *Class::*SetGrpsFrozen(*short* value);

Visual Basic [form.]control.GrpsFrozen[= value%]

Designer Page

[General subtab of the Groups designer page](#)

Remarks

The default value for the GrpsFrozen property is 0.

Setting the GrpsFrozen property to 1 freezes the first group (group 0, the top, leftmost group) so that it remains visible when the user clicks the horizontal scroll bar. Setting the GrpsFrozen property to values greater than 1 freezes the corresponding groups. For example, setting the GrpsFrozen property to 2 freezes the first and second groups.

Before setting the GrpsFrozen property, you must create a multiple-group fpCombo or fpList control by setting the [Groups](#) property.

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example creates a combo box control with four columns and four groups. The first three groups are frozen to prevent horizontal scrolling

C

```
LC_SetColumns(hWnd, 4);
LC_SetColumnHeaderShow(hWnd, FALSE);
LC_SetGroups(hWnd, 4);
LC_SetGroupHeaderShow(hWnd, TRUE);
/* Freeze first three groups */
LC_SetGrpsFrozen(hWnd, 3);
/* Define groups */
LC_SetGrp(hWnd, 0);
fpCombo1->SetGrpHeaderText(hWnd, "Name");
LC_SetGrp(hWnd, 1);
LC_SetGrpHeaderText(hWnd, "Address");
LC_SetGrp(hWnd, 2);
LC_SetGrpHeaderText(hWnd, "Phone");
LC_SetGrp(hWnd, 3);
LC_SetGrpHeaderText(hWnd, "E-Mail");
/* Assign cols to groups */
LC_SetCol(hWnd, 0);
LC_SetColParentGroup(hWnd, 0);
LC_SetCol(hWnd, 1);
LC_SetColParentGroup(hWnd, 1);
LC_SetCol(hWnd, 2);
LC_SetColParentGroup(hWnd, 2);
LC_SetCol(hWnd, 3);
LC_SetColParentGroup(hWnd, 3);
```

C++

```
fpCombo1->SetColumns(4);
fpCombo1->SetColumnHeaderShow(FALSE);
fpCombo1->SetGroups(4);
fpCombo1->SetGroupHeaderShow(TRUE);
// Freeze first three groups
fpCombo1->SetGrpsFrozen(3);
// Define groups
fpCombo1->SetGrp(0);
fpCombo1->SetGrpHeaderText("Name");
fpCombo1->SetGrp(1);
fpCombo1->SetGrpHeaderText("Address");
fpCombo1->SetGrp(2);
fpCombo1->SetGrpHeaderText("Phone");
fpCombo1->SetGrp(3);
fpCombo1->SetGrpHeaderText("E-Mail");
// Assign cols to groups
fpCombo1->SetCol(0);
fpCombo1->SetColParentGroup(0);
fpCombo1->SetCol(1);
fpCombo1->SetColParentGroup(1);
fpCombo1->SetCol(2);
fpCombo1->SetColParentGroup(2);
fpCombo1->SetCol(3);
fpCombo1->SetColParentGroup(3);
```

Visual Basic

```
fpCombo1.Columns = 4
fpCombo1.ColumnHeaderShow = False
fpCombo1.Groups = 4
fpCombo1.GroupHeaderShow = True
' Freeze first three groups
fpCombo1.GrpsFrozen = 3
```

```
' Define groups
fpCombo1.Grp = 0
fpCombo1.GrpHeaderText = "Name"
fpCombo1.Grp = 1
fpCombo1.GrpHeaderText = "Address"
fpCombo1.Grp = 2
fpCombo1.GrpHeaderText = "Phone"
fpCombo1.Grp = 3
fpCombo1.GrpHeaderText = "E-Mail"
' Assign cols to groups
fpCombo1.Col = 0
fpCombo1.ColParentGroup = 0
fpCombo1.Col = 1
fpCombo1.ColParentGroup = 1
fpCombo1.Col = 2
fpCombo1.ColParentGroup = 2
fpCombo1.Col = 3
fpCombo1.ColParentGroup = 3
```

See Also

[Customizing Groups](#)

[Groups](#), [Grp](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

GrpWidth Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the width of a group.

Syntax

C **UINT** LC_GetGrpWidth(**HWND** hWnd, **float FAR** *lpValue);
UINT LC_SetGrpWidth(**HWND** hWnd, **float** value);

C++ **float** **Class**::GetGrpWidth(**void**);
Class::SetGrpWidth(**float** value);

Visual Basic [form.]control.GrpWidth[= value!]

Designer Page

[Specific subtab of the Groups designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the GrpWidth property is 1, which specifies that the group width is automatically sized to the default column width. The unit of measurement is determined by the [ColumnWidthScale](#) property.

If you group groups, widths for groups are adjusted to fit within the parent group and the GrpWidth property setting is ignored. For more information, see [Calculating the Width of Group Children](#).

Before you set the GrpWidth property, you must specify a column with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Columns that are children of a group are automatically sized to fit the group width. The GrpWidth property overrides the [ColWidth](#) property.

Data Type

Single

[Print](#)

[Copy](#)

[Close](#)

The following example creates a combo box control with two groups. The width of the first group is 20 and the width of the second group is 30.

C

```
LC_SetGroups(hWnd, 2);
LC_SetGroupHeaderShow(hWnd, TRUE);
/* Define groups */
LC_SetGrp(hWnd, 0);
LC_SetGrpWidth(hWnd, 20);
LC_SetGrpHeaderText(hWnd, "Date of Service");
LC_SetGrp(hWnd, 1);
LC_SetGrpWidth(hWnd, 30);
LC_SetGrpHeaderText(hWnd, "Service Provided By");
```

C++

```
fpCombo1->SetGroups(2);
fpCombo1->SetGroupHeaderShow(TRUE);
// Define groups
fpCombo1->SetGrp(0);
fpCombo1->SetGrpWidth(20);
fpCombo1->SetGrpHeaderText("Date of Service");
fpCombo1->SetGrp(1);
fpCombo1->SetGrpWidth(30);
fpCombo1->SetGrpHeaderText("Service Provided By");
```

Visual Basic

```
fpCombo1.Groups = 2
fpCombo1.GroupHeaderShow = True
' Define groups
fpCombo1.Grp = 0
fpCombo1.GrpWidth = 20
fpCombo1.GrpHeaderText = "Date of Service"
fpCombo1.Grp = 1
fpCombo1.GrpWidth = 30
fpCombo1.GrpHeaderText = "Service Provided By"
```

See Also

[Calculating the Width of Group Children](#)

[Specifying the Group Width](#)

[ColumnWidthScale](#), [ColWidth](#), [Groups](#), [Grp](#), [GrpFromID](#), [GrpFromName](#), [GrpID](#), [GrpLockResize](#), [GrpName](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

HighestPrecedence Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns which color is displayed at the intersection of a row and column with color settings.

Syntax

C **UINT** LC_GetHighestPrecedence(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetHighestPrecedence(**HWND** hWnd, **short** value);

C++ **short Class::**GetHighestPrecedence(**void**);
Class::SetHighestPrecedence(**short** value);

Visual Basic [form.]control.HighestPrecedence[= setting%]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

When a color is set for a row and a different color is set for a column, the HighestPrecedence property determines which color takes precedence when the colors intersect.

Setting	Description	Constant
0 - Row	(Default) The row color takes precedence	LC_HIGHESTPRECEDENCE_ROW
1 - Col	The column color takes precedence	LC_HIGHESTPRECEDENCE_COL
2 - Combined	The row and column colors are combined and displayed as a third color	LC_HIGHESTPRECEDENCE_COMBINED

The HighestPrecedence property setting also influences how properties are applied to the control. For more information, see Appendix C, "Hierarchy of Property Settings" in the printed *List Pro User's Guide*.

Data Type

Integer (Enumerated)

See Also

[Applying Properties to Specific Parts of the Control](#)

[BackColor](#), [ForeColor](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
	✓		✓	

hWnd Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Returns the window handle of the control. This property is available for ActiveX and VBX controls only.

Syntax

C++ **short Class::GethWnd(void);**

Visual Basic [*form.*][*control.*]hWnd

ActiveX Use

The hWnd property is a stock property.

Remarks

Use the hWnd property with Windows API calls.

Refer to the Visual Basic documentation for additional information about this property.

Data Type

Integer

See Also

[DataSourceWnd](#), [DataSourceWndList](#) properties

PD	RD	WR	RT	DT
✓		✓	✓	

InsertRow Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets whether to insert a new item or row of text in the list of an fpCombo or fpList control.

Syntax

C **UINT** LC_SetInsertRow(**HWND** hWnd, **LPCSTR** value);

C++ **Class::**SetInsertRow(**LPCSTR** value);

Visual Basic [form.]control.InsertRow[= text\$]

Designer Page

[Add Data designer page](#)

Remarks

The InsertRow property is functionally equivalent to the Visual Basic AddItem method. However, the fpCombo and fpList controls can display more rows than the AddItem method can add (the AddItem method is limited to 32,768 rows). By using the InsertRow property, you can add two billion rows to an fpCombo or fpList control.

By default (when the [Row](#) property is set to its default value, 0), inserting items with the InsertRow property adds a new row to

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

the top of the list. If you set the Row property to 1, new rows are added to the end of the list. If the list is sorted, set the Row property to

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

1, otherwise the rows will be inserted in an unsorted manner.

Data Type

String

See Also

[Adding List Items](#)

[Row](#) property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ItemData Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns a long integer value that is associated with a row in the list.

Syntax

C *UINT* LC_GetItemData(*HWND* hWnd, *long FAR* *pValue);
 UINT LC_SetItemData(*HWND* hWnd, *long* value);

C++ *long* *Class*::GetItemData(*void*);
 Class::SetItemData(*long* value);

Visual Basic [form.]control.ItemData[= value&]

Remarks

Use the ItemData property to associate unique four-byte values to rows in an fpCombo or fpList control. You can then use these

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

numbers in code to identify the items. If you do not specify a row (that is, the value of the [Row](#) property is 1) before setting this property, the ItemData property will return an error code value of 30501.

A typical use of the ItemData property is for an index into an array of data structures associated with items in an fpList control.

For example, you can use an employee's identification number to identify each employee name in an fpList control. By writing appropriate code, you can sort the numbers or let users search the numbers. To insert the identification numbers, fill the corresponding elements in the ItemData array with the employee numbers.

Data Type

Integer (Long)

[Print](#)

[Copy](#)

[Close](#)

The following example adds items to an fpCombo control. The code assigns an identification number to each item using the ItemData and [NewIndex](#) properties.

C

```
LC_SetColumns(hWnd, 1);
LC_SetColumnHeaderShow(hWnd, FALSE);
LC_SetRow(hWnd, -1);
LC_SetInsertRow(hWnd, "John Doe");
LC_SetRow(hWnd, LC_GetNewIndex(hWnd, &xIndex));
LC_SetItemData(hWnd, "123456789");
LC_SetInsertRow(hWnd, "Mary Smith");
LC_SetRow(hWnd, LC_GetNewIndex(hWnd, &xIndex));
LC_SetItemData(hWnd, "987654321");
```

C++

```
fpCombo1->SetColumns(1);
fpCombo1->SetColumnHeaderShow(FALSE);
fpCombo1->SetRow(-1);
fpCombo1->SetInsertRow("John Doe");
fpCombo1->SetRow(fpCombo1->GetNewIndex( ));
fpCombo1->SetItemData("123456789");
fpCombo1->SetInsertRow("Mary Smith");
fpCombo1->SetRow(fpCombo1->GetNewIndex( ));
fpCombo1->SetItemData("987654321");
```

Visual Basic

```
fpCombo1.Columns = 1
fpCombo1.ColumnHeaderShow = False
fpCombo1.Row = -1
fpCombo1.InsertRow = "John Doe"
fpCombo1.Row = fpCombo1.NewIndex
fpCombo1.ItemData = "123456789"
fpCombo1.InsertRow = "Mary Smith"
fpCombo1.Row = fpCombo1.NewIndex
fpCombo1.ItemData = "987654321"
```

See Also

[NewIndex](#), [Row](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

JoinID Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the identification number for a joined set of cells.

Syntax

C **UINT** LC_GetJoinID(*HWND* hWnd, **long FAR** *lpValue);
UINT LC_SetJoinID(*HWND* hWnd, **long** value);

C++ **long** *Class*::GetJoinID(*void*);
Class::SetJoinID(**long** value);

Visual Basic [*form.*]*control*.JoinID[= *value*&]

Designer Page

[Merge/Join designer page](#)

Remarks

With the JoinID property, you can create a set of cells that share property characteristics (that is, they are identical in content and appearance). The property characteristics of the first cell you assign to the joined set are applied to the other cells in the joined set. Adjacent cells with the same identification number are also merged unless rows have multiple levels. Adjacent cells in rows with multiple levels have the same content and appearance but are not merged.

The advantage of joined cells is that a property that is set for one joined cell is applied to all joined cells. For example, if you want the same background color for all cells in a joined set, you set the [BackColor](#) property for one cell in the set.

All cells on one form with the same identification number are in the same joined set. The identification number for a cell must be greater than 0 for the cell to be considered a part of the joined set. Joined cells do not have to be adjacent to one another.

The default value for the JoinID property is 0, which specifies that the cell is not part of a joined set.

You must set the [Col](#) and [Row](#) properties to define a cell before setting this property.

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example creates a list box control with three columns. Cells in the first two columns in the first four rows are joined.

C

```
LC_SetColumns(hWnd, 3);
LC_SetColumnHeaderShow(hWnd, FALSE);
LC_SetColumnLevels(hWnd, 2);
LC_SetLineStyle(hWnd, LC_LISTSTYLE_LOWERED);
for (j = 0; j < 3; j++)
{
    LC_SetRow(hWnd, j);
    for (i = 0; i < 3; i++)
    {
        LC_SetCol(hWnd, i);
        LC_SetColText(hWnd, i);
    }
}

LC_SetCol(hWnd, 1);
LC_SetColLevel(hWnd, 1);
for (k = 0; k < 3; k++)
    LC_SetCol(hWnd, 0);
LC_SetRow(hWnd, k);
LC_SetJoinID(hWnd, 18);
LC_SetCol(hWnd, 1);
LC_SetRow(hWnd, k);
LC_SetJoinID(hWnd, 18);
```

C++

```
fpList1->SetColumns(3);
fpList1->SetColumnHeaderShow(FALSE);
fpList1->SetColumnLevels(2);
fpList1->SetLineStyle(LC_LISTSTYLE_LOWERED);
for (j = 0; j < 3; j++)
{
    fpList1->SetRow(j);
    for (i = 0; i < 2; i++)
    {
        fpList1->SetCol(i);
        fpList1->SetColText(i);
    }
}

fpList1->SetCol(1);
fpList1->SetColLevel(1);
for (k = 0; k < 3; k++)
    fpList1->SetCol(0);
fpList1->SetRow(k);
fpList1->SetJoinID(18);
fpList1->SetCol(1);
fpList1->SetRow(k);
fpList1->SetJoinID(18);
```

Visual Basic

```
fpList1.Columns = 3
fpList1.ColumnHeaderShow = False
fpList1.ColumnLevels = 2
fpList1.LineStyle = LC_LISTSTYLE_LOWERED
For j = 0 to 3
    fpList1.Row = j
    For I = 0 to 2
        fpList1.Col = I
```

```
        fpList1.ColText = I
    Next
Next

fpList1.Col = 1
fpList1.ColLevel = 1
For k = 0 to 3
    fpList1.Col = 0
    fpList1.Row = k
    fpList1.JoinID = 18
    fpList1.Col = 1
    fpList1.Row = k
    fpList1.JoinID = 18
Next
```

See Also

[Joining Cells](#)

[Col](#), [Row](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Line3DDark Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the shadow color of three-dimensional lines.

Syntax

C `UINT LC_GetLine3DDark(HWND hWnd, COLORREF FAR *lpValue);
UINT LC_SetLine3DDark(HWND hWnd, COLORREF value);`

C++ `COLORREF Class::GetLine3DDark(void);
Class::SetLine3DDark(COLORREF value);`

Visual Basic `[form.]control.Line3DDark[= color]`

Designer Page

[Line subtab of the ApplyTo designer page](#)

Remarks

The default value for the Line3DDark property is &H80000010& (Windows system button shadow color).

Use the Line3DDark and [Line3DLight](#) properties to change the three-dimensional line colors. To create three-dimensional lines along the borders of each cell, you must set the [LineStyle](#) property to either 3 (Lowered), 4 (Raised), 5 (Lowered w/ Line), or 6 (Raised w/ Line).

You can set the [ListApplyTo](#) property before you set the Line3DDark property to designate the part of the control to which the three-dimensional shadow color applies. Use the [LineApplyTo](#) property to specify the lines to which the shadow color applies. Use the [Line3DWidth](#) property to specify the line width.

Data Type

Color

See Also

[Customizing Lines](#)

[Line3DLight](#), [Line3DWidth](#), [LineApplyTo](#), [LineStyle](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Line3DLight Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the highlight color of three-dimensional lines.

Syntax

C	UINT LC_GetLine3DLight(HWND hWnd, COLORREF FAR *pValue); UINT LC_SetLine3DLight(HWND hWnd, COLORREF value);
C++	COLORREF Class ::GetLine3DLight(void); Class ::SetLine3DLight(COLORREF value);
Visual Basic	[form.]control.Line3DLight[= color]

Designer Page

[Line subtab of the ApplyTo designer page](#)

Remarks

The default value for the Line3DLight property is &H00FFFFFF& (white).

Use the [Line3DDark](#) and Line3DLight properties to change the three-dimensional line colors. To create three-dimensional lines along the borders of each cell, you must set the [LineStyle](#) property to either 3 (Lowered), 4 (Raised), 5 (Lowered w/ Line), or 6 (Raised w/ Line).

You can set the [ListApplyTo](#) property before you set the Line3DLight property to designate the part of the control to which the three-dimensional highlight color applies. Use the [LineApplyTo](#) property to specify the lines to which the highlight color applies. Use the [Line3DWidth](#) property to specify the line width.

Data Type

Color

See Also

[Customizing Lines](#)

[Line3DDark](#), [Line3DWidth](#), [LineApplyTo](#), [LineStyle](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Line3DWidth Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the width in pixels of three-dimensional lines.

Syntax

C **UINT** LC_GetLine3DWidth(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetLine3DWidth(**HWND** hWnd, **short** value);

C++ **short** **Class**::GetLine3DWidth(**void**);
Class::SetLine3DWidth(**short** value);

Visual Basic [form.]control.Line3DWidth[= value%]

Designer Page

[Line subtab of the ApplyTo designer page](#)

Remarks

The default value for the Line3DWidth property is 1 pixel.

To create three-dimensional lines along the right and bottom borders of each cell, you must set the [LineStyle](#) property to either 3 (Lowered), 4 (Raised), 5 (Lowered w/ Line), or 6 (Raised w/ Line). Use the [Line3DDark](#) and [Line3DLight](#) properties to change the three-dimensional line colors.

You can set the [ListApplyTo](#) property before you set the Line3DWidth property to designate the part of the control to which the three-dimensional width applies. Use the [LineApplyTo](#) property to specify the lines to which the width applies.

Both the highlight and shadow portions are set to the width specified by the Line3DWidth property. Therefore, the total width of three-dimensional lines is equal to the sum of the widths of the highlight and shadow portions.

Use the [LineWidth](#) property to specify the width of flat lines.

Data Type

Integer

See Also

[Customizing Lines](#)

[Line3DDark](#), [Line3DLight](#), [LineApplyTo](#), [LineStyle](#), [LineWidth](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

LineApplyTo Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether line properties apply to the lines between rows, between columns, or between columns when multiple levels exist, or to all lines.

Syntax

C *UINT* LC_GetLineApplyTo(*HWND* hWnd, *short FAR* *pValue);
UINT LC_SetLineApplyTo(*HWND* hWnd, *short* value);

C++ *short Class*::GetLineApplyTo(*void*);
Class::SetLineApplyTo(*short* value);

Visual Basic [form.]control.LineApplyTo[= setting%]

Designer Page

Line Apply To drop-down list box on the [Line subtab of the ApplyTo designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Default	(Default) Applies properties to all lines	LC_LINEAPPLYTO_DEFAULT
1 - Rows	Applies properties to lines between rows	LC_LINEAPPLYTO_ROWS
2 - Cols	Applies properties to lines between columns	LC_LINEAPPLYTO_COLS
3 - ColsH	Applies properties to horizontal lines between columns when multiple levels exist	LC_LINEAPPLYTO_COLSH

You can set the [ListApplyTo](#) property before you set the LineApplyTo property to designate the part of the control to which this property applies.

Use the LineApplyTo property before setting the following line properties:

Line3DDark	LineColor
Line3DLight	LineStyle
Line3DWidth	LineWidth

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a list box control that displays a lowered, three-dimensional line style. The width of the three-dimensional lines is 2 pixels. When a row is selected, a 3-pixel, blue, flat line is displayed below the row.

C

```
/* Display lowered line style */
LC_SetLineStyle(hWnd, LC_LISTSTYLE_LOWERED);
/* green RGB(0, 255, 0) */
LC_SetLine3DDark(hWnd, 0x0000FF00);
/* red RGB(255, 0, 0) */
LC_SetLine3DLight(hWnd, 0x000000FF);
LC_SetLine3DWidth(hWnd, 2);
/* Display flat line when row is selected */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SEL_ROWS);
LC_SetLineApplyTo(hWnd, LC_LINEAPPLYTO_ROWS);
LC_SetLineStyle(hWnd, LC_LISTSTYLE_FLAT);
LC_SetLineWidth(hWnd, 3);
/* blue RGB(0, 0, 255) */
LC_SetLineColor(hWnd, 0x00FF0000);
```

C++

```
// Display lowered line style
fpList1->SetLineStyle(LC_LISTSTYLE_LOWERED);
// green RGB(0, 255, 0)
fpList1->SetLine3DDark(0x0000FF00);
// red RGB(255, 0, 0)
fpList1->SetLine3DLight(0x000000FF);
fpList1->SetLine3DWidth(2);
// Display flat line when row is selected
fpList1->SetListApplyTo(LC_LISTAPPLYTO_SEL_ROWS);
fpList1->SetLineApplyTo(LC_LINEAPPLYTO_ROWS);
fpList1->SetLineStyle(LC_LISTSTYLE_FLAT);
fpList1->SetLineWidth(3);
// blue RGB(0, 0, 255)
fpList1->SetLineColor(0x00FF0000);
```

Visual Basic

```
' Display lowered line style
fpList1.LineStyle = LC_LISTSTYLE_LOWERED
' green RGB(0, 255, 0)
fpList1.Line3DDark = &H0000FF00&
' red RGB(255, 0, 0)
fpList1.Line3DLight = &H000000FF&
fpList1.Line3DWidth = 2
' Display flat line when row is selected
fpList1.ListApplyTo = LC_LISTAPPLYTO_SEL_ROWS
fpList1.LineApplyTo = LC_LINEAPPLYTO_ROWS
fpList1.LineStyle = LC_LISTSTYLE_FLAT
fpList1.LineWidth = 3
' blue RGB(0, 0, 255)
fpList1.LineColor = &H00FF0000&
```

See Also

[Working with Lines](#)

[Line3DDark](#), [Line3DLight](#), [Line3DWidth](#), [LineColor](#), [LineStyle](#), [LineWidth](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

LineColor Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the color of flat lines between columns and rows.

Syntax

C *UINT* LC_GetLineColor(*HWND* hWnd, *COLORREF FAR* *lpValue);
 UINT LC_SetLineColor(*HWND* hWnd, *COLORREF* value);

C++ *COLORREF* *Class*::GetLineColor(*void*);
 Class::SetLineColor(*COLORREF* value);

Visual Basic [form.]control.LineColor[= color]

Designer Page

[Line subtab of the ApplyTo designer page](#)

Remarks

The default value for the LineColor property is &H80000008& (Windows system text color).

Set the [LineStyle](#) property to 2 (Flat), 5 (Lowered w/ Line), or 6 (Raised w/ Line) to display flat lines.

You can set the [ListApplyTo](#) property before you set the LineStyle property to designate the part of the control to which the line style applies. Use the [LineApplyTo](#) property to specify the lines to which the line color applies.

Use the [Line3DDark](#) and [Line3DLight](#) properties to change the highlight and shadow colors of three-dimensional lines.

Data Type

Color

See Also

[Customizing Lines](#)

[Line3DDark](#), [Line3DLight](#), [LineApplyTo](#), [LineStyle](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

LineStyle Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the appearance of lines between columns, rows, or both.

Syntax

C **UINT** LC_GetLineStyle(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetLineStyle(**HWND** hWnd, **short** value);

C++ **short** *Class*::GetLineStyle(**void**);
Class::SetLineStyle(**short** value);

Visual Basic [form.]control.LineStyle[= setting%]

Designer Page

[Line subtab of the ApplyTo designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Default	(Default) Uses the line style of the hierarchical predecessor	LC_LINESTYLE_DEFAULT
1 - None	Does not display lines	LC_LINESTYLE_NONE
2 - Flat	Displays a flat line	LC_LINESTYLE_FLAT
3 - Lowered	Displays lines with a lowered three-dimensional appearance	LC_LINESTYLE_LOWERED
4 - Raised	Displays lines with a raised three-dimensional appearance	LC_LINESTYLE_RAISED
5 - Lowered w/ Line	Displays lines with a lowered three-dimensional appearance and a flat line	LC_LINESTYLE_LOWERED_W_LINE
6 - Raised w/ Line	Displays lines with a raised three-dimensional appearance and a flat line	LC_LINESTYLE_RAISED_W_LINE

If you set the LineStyle property to 2 (Flat), 5 (Lowered w/ Line), or 6 (Raised w/ Line), you can set the [LineApplyTo](#) property and designated-line properties to define where and how lines are displayed.

You can set the [ListApplyTo](#) property before you set the LineStyle property to designate the part of the control to which the line style applies.

Data Type

Integer (Enumerated)

See Also

[Working with Lines](#)

[ExtendCol](#), [ExtendRow](#), [LineApplyTo](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

LineWidth Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the width in pixels of flat lines.

Syntax

C	UINT LC_GetLineWidth(HWND hWnd, short FAR *lpValue); UINT LC_SetLineWidth(HWND hWnd, short value);
C++	short Class:: GetLineWidth(void); Class:: SetLineWidth(short value);
Visual Basic	[form.]control.LineWidth[=value%]

Designer Page

[Line subtab of the ApplyTo designer page](#)

Remarks

The default value for the LineWidth property is 1 pixel.

To create flat lines, set the [LineStyle](#) property to 2 (Flat), 5 (Lowered w/ Line), or 6 (Raised w/ Line).

You can set the [ListApplyTo](#) property before you set the LineWidth property to designate the part of the control to which the line width applies. Use the [LineApplyTo](#) property to specify the lines to which the width applies.

Use the [Line3DWidth](#) property to specify the width of three-dimensional lines.

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example creates a single-column fpList control with three-dimensional lines for both the list and the selections and sorts the values in ascending order.

C

```
LC_SetSorted(hWnd, LC_SORTED_ASCENDING);
LC_SetRow(hWnd, -1);
LC_SetInsertRow(hWnd, "Ocracoke");
LC_SetInsertRow(hWnd, "Chowan");
LC_SetInsertRow(hWnd, "Hatteras");
LC_SetInsertRow(hWnd, "Kill Devil Hills");
LC_SetInsertRow(hWnd, "Kitty Hawk");
LC_SetInsertRow(hWnd, "Croatan");
LC_SetInsertRow(hWnd, "Oregon Inlet");
/* Create a 3D look for the list box */
/* light gray RGB(192, 192, 192) */
LC_SetBackColor(hWnd, 0x00C0C0C0);
LC_SetLineApplyTo(hWnd, LC_LINEAPPLYTO_DEFAULT);
LC_SetLineStyle(hWnd, LC_LINESTYLE_LOWERED_W_LINE);
/* white RGB(255, 255, 255) */
LC_SetLine3DLight(hWnd, 0x00FFFFFF);
/* dark gray RGB(128, 128, 128) */
LC_SetLine3DDark(hWnd, 0x00808080);
LC_SetLineWidth(hWnd, 3);
/* Create a 3D look for the selections (allow up to 5 selections) */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SEL_ROWS);
LC_SetLineStyle(hWnd, LC_LINESTYLE_RAISED);
/* medium blue RGB(0, 0, 192) */
LC_SetLine3DLight(hWnd, 0x00C00000);
/* dark blue RGB(0, 0, 128) */
LC_SetLine3DDark(hWnd, 0x00800000);
LC_SetLine3DWidth(hWnd, 2);
LC_SetSelDrawFocusRect(hWnd, FALSE);
LC_SetSelMax(hWnd, 5);
```

C++

```
fpList1->SetSorted(LC_SORTED_ASCENDING);
fpList1->SetRow(-1);
fpList1->SetInsertRow("Ocracoke");
fpList1->SetInsertRow("Chowan");
fpList1->SetInsertRow("Hatteras");
fpList1->SetInsertRow("Kill Devil Hills");
fpList1->SetInsertRow("Kitty Hawk");
fpList1->SetInsertRow("Croatan");
fpList1->SetInsertRow("Oregon Inlet");
// Create a 3D look for the list box
// light gray RGB(192, 192, 192)
fpList1->SetBackColor(0x00C0C0C0&);
fpList1->SetLineApplyTo(LC_LINEAPPLYTO_DEFAULT);
fpList1->SetLineStyle(LC_LINESTYLE_LOWERED_W_LINE);
// white RGB(255, 255, 255)
fpList1->SetLine3DLight(0x00FFFFFF);
// dark gray RGB(128, 128, 128)
fpList1->SetLine3DDark(0x00808080);
fpList1->SetLineWidth(3);
// Create a 3D look for the selections (allow up to 5 selections)
fpList1->SetListApplyTo(LC_LISTAPPLYTO_SEL_ROWS);
fpList1->SetLineStyle(LC_LINESTYLE_RAISED);
// medium blue RGB(0, 0, 192)
fpList1->SetLine3DLight(0x00C00000);
// dark blue RGB(0, 0, 128)
fpList1->SetLine3DDark(0x00800000&);
fpList1->SetLine3DWidth(2);
```

```
fpList1->SetSelDrawFocusRect (FALSE);  
fpList1->SetSelMax(5);
```

Visual Basic

```
fpList1.Sorted = LC_SORTED_ASCENDING  
fpList1.Row = -1  
fpList1.InsertRow = "Ocracoke"  
fpList1.InsertRow = "Chowan"  
fpList1.InsertRow = "Hatteras"  
fpList1.InsertRow = "Kill Devil Hills"  
fpList1.InsertRow = "Kitty Hawk"  
fpList1.InsertRow = "Croatan"  
fpList1.InsertRow = "Oregon Inlet"  
' Create a 3D look for the list box  
' light gray RGB(192, 192, 192)  
fpList1.BackColor = &H00C0C0C0&  
fpList1.LineApplyTo = LC_LINEAPPLYTO_DEFAULT  
fpList1.LineStyle = LC_LINESTYLE_LOWERED_W_LINE  
' white RGB(255, 255, 255)  
fpList1.Line3DLight = &H00FFFFFF&  
' dark gray RGB(128, 128, 128)  
fpList1.Line3DDark = &H00808080&  
fpList1.LineWidth = 3  
' Create a 3D look for the selections (allow up to 5 selections)  
fpList1.ListApplyTo = LC_LISTAPPLYTO_SEL_ROWS  
fpList1.LineStyle = LC_LINESTYLE_RAISED  
' medium blue RGB(0, 0, 192)  
fpList1.Line3DLight = &H00C00000&  
' dark blue RGB(0, 0, 128)  
fpList1.Line3DDark = &H00800000&  
fpList1.Line3DWidth = 2  
fpList1.SelDrawFocusRect = False  
fpList1.SelMax = 5
```

See Also

[Customizing Lines](#)

[Line3DWidth](#), [LineApplyTo](#), [LineStyle](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

List Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns text from one row in the list portion of an fpCombo or fpList control. This property is available at run time only.

Syntax

C	UINT LC_GetList(HWND hWnd, LPSTR buffer, UINT nBufferSize); UINT LC_SetList(HWND hWnd, LPCSTR value);
C++	LPSTR Class ::GetList(LPSTR buffer, UINT bufferSize); Class ::SetList(LPCSTR value);
Visual Basic	[form.]control.List[=text\$]

Designer Page

[Add Data designer page](#)

Remarks

Use the List property to access list items in single-column fpCombo and fpList controls (when the [Columns](#) property is set to 0).

If you use the List property to access list items in multiple-column fpCombo and fpList controls, the column values are separated by the character specified by the [ColumnSeparatorChar](#) property. Use the [ColList](#) property to set or return specific column values in multiple-column fpCombo and fpList controls.

You must specify a row with the [Row](#) property before you set the List property.

Note The List property is not an array property as it is in Visual Basic.

Data Type

String

See Also

[Accessing List Items](#)

[ColList](#), [Columns](#), [ColumnSeparatorChar](#), [Row](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

List3DText Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether text in an fpCombo or fpList control looks three-dimensional.

Syntax

C **UINT** LC_GetList3DText(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetList3DText(**HWND** hWnd, **short** value);

C++ **short Class::**GetList3DText(**void**);
Class::SetList3DText(**short** value);

Visual Basic [form.]control.List3DText[= setting%]

Designer Page

[List subtab of the ApplyTo designer page](#)

Remarks

The List3DText property adds three-dimensional effects to the text in the list.

The following settings are available:

Setting	Description	Constants
0 - None	(Default) Removes three-dimensional effect from text	LC_LIST3DTEXT_NONE
1 - Lowered	Creates lowered three-dimensional text	LC_LIST3DTEXT_LOWERED
2 - Raised	Creates raised three-dimensional text	LC_LIST3DTEXT_RAISED
3 - Lowered w/ Shading	Creates lowered three-dimensional text with extra shading	LC_LIST3DTEXT_LOWERED_W_SHADING
4 - Raised w/ Shading	Creates raised three-dimensional text with extra shading	LC_LIST3DTEXT_RAISED_W_SHADING
5 - Default	Uses the three-dimensional text effect of the hierarchical predecessor	LC_LIST3DTEXT_DEFAULT

You can set the [ListApplyTo](#) property before you set the List3DText property to designate the part of the control to which the three-dimensional text effect applies.

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates an fpList control that contains a list of the names of rivers. The text in the list is three-dimensional, and the colors of the background and the text of an item change when the user selects the item.

C

```
LC_SetRow(hWnd, -1);
LC_SetInsertRow(hWnd, "Nile");
LC_SetInsertRow(hWnd, "Ganges");
LC_SetInsertRow(hWnd, "Mississippi");
LC_SetInsertRow(hWnd, "Yangtze");
LC_SetInsertRow(hWnd, "Orinoco");
/* Set list text to be three-dimensional */
LC_SetList3DText(hWnd, LC_LIST3DTEXT_LOWERED_W_SHADING);
LC_SetList3DTextOffset(hWnd, 1);
/* blue RGB(0, 0, 255) */
LC_SetList3DTextHighlightColor(hWnd, 0x00FF0000);
/* light gray RGB(192, 192, 192) */
LC_SetList3DTextShadowColor(hWnd, 0x00C0C0C0);
/* Set background and text colors of selected items */
LC_SetListApplyTo(hWnd, LC_LISTAPPLY_SEL_ROWS);
/* green RGB(0, 255, 0) */
LC_SetBackColor(hWnd, 0x0000FF00);
/* red RGB(255, 0, 0) */
LC_SetForeColor(hWnd, 0x000000FF);
```

C++

```
fpList1->SetRow(-1);
fpList1->SetInsertRow("Nile");
fpList1->SetInsertRow("Ganges");
fpList1->SetInsertRow("Mississippi");
fpList1->SetInsertRow("Yangtze");
fpList1->SetInsertRow("Orinoco");
// Set list text to be three-dimensional
fpList1->SetList3DText(LC_LIST3DTEXT_LOWERED_W_SHADING);
fpList1->SetList3DTextOffset(1);
// blue RGB(0, 0, 255)
fpList1->SetList3DTextHighlightColor(0x00FF0000);
// light gray RGB(192, 192, 192)
fpList1->SetList3DTextShadowColor(0x00C0C0C0);
// Set background and text colors of selected items
fpList1->SetListApplyTo(LC_LISTAPPLYTO_SEL_ROWS);
// green RGB(0, 255, 0)
fpList1->SetBackColor(0x0000FF00);
// red RGB(255, 0, 0)
fpList1->SetForeColor(0x000000FF);
```

Visual Basic

```
fpList1.Row = -1
fpList1.InsertRow = "Nile"
fpList1.InsertRow = "Ganges"
fpList1.InsertRow = "Mississippi"
fpList1.InsertRow = "Yangtze"
fpList1.InsertRow = "Orinoco"
' Set list text to be three-dimensional
fpList1.List3DText = LC_LIST3DTEXT_LOWERED_W_SHADING
fpList1.List3DTextOffset = 1
' blue RGB(0, 0, 255)
fpList1.List3DTextHighlightColor = &H00FF0000&
' light gray RGB(192, 192, 192)
fpList1.List3DTextShadowColor = &H00C0C0C0&
' Set background and text colors of selected items
fpList1.ListApplyTo = LC_LISTAPPLYTO_SEL_ROWS
' green RGB(0, 255, 0)
```

```
fpList1.BackColor = &H0000FF00&  
' red RGB(255, 0, 0)  
fpList1.ForeColor = &H000000FF&
```

See Also

[Creating Three-Dimensional Text](#)

[List3DTextHighlightColor](#), [List3DTextOffset](#), [List3DTextShadowColor](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

List3DTextHighlightColor, List3DTextShadowColor Properties

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Set or return the highlight and shadow colors of the three-dimensional text in an fpCombo or fpList control.

Syntax

C **UINT** LC_GetList3DTextHighlightColor(**HWND** hWnd, **COLORREF FAR** *lpValue);
UINT LC_SetList3DTextHighlightColor(**HWND** hWnd, **COLORREF** value);

C++ **COLORREF** *Class*::GetList3DTextHighlightColor(**void**);
Class::SetList3DTextHighlightColor(**COLORREF** value);

Visual Basic [form.]control.List3DTextHighlightColor[= color]

Note The List3DTextHighlightColor and List3DTextShadowColor properties use the same syntax.

Designer Page

[List subtab of the ApplyTo designer page](#)

Remarks

The List3DTextHighlightColor and List3DTextShadowColor properties let you change the highlight and shadow colors of three-dimensional text. To create three-dimensional text, set the [List3DText](#) property to a value other than 0 (None). To heighten the three-dimensional effect, choose a light shade for the highlight color and a darker shade for the shadow color. For the List3DTextShadowColor property to have an effect, the List3DText property must be set to 3 (Lowered w/ Shading) or 4 (Raised w/ Shading).

The default values for these properties are &H80000014& (Windows system button highlight color) for List3DTextHighlightColor and &H80000010& (Windows system button shadow color) for List3DTextShadowColor.

You can set the [ListApplyTo](#) property before you set the List3DTextHighlightColor or List3DTextShadowColor property to designate the part of the control to which the three-dimensional highlight or shadow color applies.

Data Type

Color

See Also

[Creating Three-Dimensional Text](#)

[List3DText](#), [List3DTextOffset](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

List3DTextOffset Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the width in pixels of the offset for three-dimensional text in an fpCombo or fpList control.

Syntax

C `UINT LC_GetList3DTextOffset(HWND hWnd, short FAR *lpValue);`
UINT LC_SetList3DTextOffset(**HWND** hWnd, **short** value);

C++ `short Class::GetList3DTextOffset(void);`
Class::SetList3DTextOffset(**short** value);

Visual Basic `[form.]control.List3DTextOffset[= value%]`

Designer Page

[List subtab of the ApplyTo designer page](#)

Remarks

Changing the List3DTextOffset property makes three-dimensional text appear more raised or lowered. As a rule, the larger the font size, the more you should increase the offset.

The default value for the List3DTextOffset property is 1 pixel.

You can set the [ListApplyTo](#) property before you set the List3DTextOffset property to designate the part of the control to which the three-dimensional text offset applies.

Data Type

Integer

See Also

[Creating Three-Dimensional Text](#)

[List3DText](#), [List3DTextHighlightColor](#), [List3DTextShadowColor](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ListApplyTo Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns where to apply designated-list properties.

Syntax

C **UINT** LC_GetListApplyTo(*HWND* hWnd, **short FAR** *lpValue);
UINT LC_SetListApplyTo(*HWND* hWnd, **short** value);

C++ **short Class::**GetListApplyTo(**void**);
Class::SetListApplyTo(**short** value);

Visual Basic [form.]control.ListApplyTo[= setting%]

Designer Pages

List Apply To drop-down list box on:

[List subtab of the ApplyTo designer page](#)

[Line subtab of the ApplyTo designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Default (All)	(Default) Applies properties to the entire control	LC_LISTAPPLYTO_DEFAULT_ALL
1 - All Rows	Applies properties to all rows	LC_LISTAPPLYTO_ALL_ROWS
2 - All Cols	Applies properties to all columns	LC_LISTAPPLYTO_ALL_COLS
3 - All Groups	Applies properties to all groups including children	LC_LISTAPPLYTO_ALL_GROUPS
4 - Sel Rows	Applies properties to selected rows	LC_LISTAPPLYTO_SEL_ROWS
5 - Odd Rows	Applies properties to odd-numbered rows	LC_LISTAPPLYTO_ODD_ROWS
6 - Even Rows	Applies properties to even-numbered rows	LC_LISTAPPLYTO_EVEN_ROWS
7 - Col Headers	Applies properties to column headers	LC_LISTAPPLYTO_COL_HEADERS
8 - Group Headers	Applies properties to group headers	LC_LISTAPPLYTO_GROUP_HEADERS
9 - Single Col Header	Applies properties to specified column header	LC_LISTAPPLYTO_SINGLE_COL_HEADER
10 - Single Group Header	Applies properties to specified group header	LC_LISTAPPLYTO_SINGLE_GROUP_HEADER
11 - Single Group	Applies properties to specified group including children	LC_LISTAPPLYTO_SINGLE_GROUP
12 - Single Item	Applies properties to a specified column, row, or cell	LC_LISTAPPLYTO_SINGLE_ITEM

If you set the ListApplyTo property to 9 (Single Col Header), set the [Col](#), [ColFromID](#), or [ColFromName](#) property to specify the column. If you set the ListApplyTo property to 10 (Single Group Header) or 11 (Single Group), set the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property to specify the group. If you set the ListApplyTo property to 12 (Single Item), set the [Col](#), [ColFromID](#), or

ColFromName property and the [Row](#) property as appropriate.

You can use the ListApplyTo property before setting any of the following properties:

AlignH	FontUnderline	List3DTextHighlightColor
AlignV	ForeColor	List3DTextOffset
BackColor	Line3DDark	List3DTextShadowColor
Font	Line3DLight	MultiLine
FontBold	Line3DWidth	Picture
FontEmpty	LineApplyTo	PictureAlignH
FontItalic	LineColor	PictureAlignV
FontName	LineStyle	PictureSel
FontSize	LineWidth	TextOrientation
FontStrikethru	List3DText	

If you set the ListApplyTo property to 3 (All Groups) or 11 (Single Group), property settings for children will be overridden.

Data Type

Integer (Enumerated)

See Also

[Applying Properties to Specific Parts of the Control](#)

[AlignH](#), [AlignV](#), [BackColor](#), [Col](#), [ColFromID](#), [ColFromName](#), [Font](#), [FontEmpty](#), [ForeColor](#), [Grp](#), [GrpFromID](#), [GrpFromName](#), [Line3DDark](#), [Line3DLight](#), [Line3DWidth](#), [LineApplyTo](#), [LineColor](#), [LineStyle](#), [LineWidth](#), [List3DText](#), [List3DTextHighlightColor](#), [List3DTextOffset](#), [List3DTextShadowColor](#), [MultiLine](#), [Picture](#), [PictureAlignH](#), [PictureAlignV](#), [PictureSel](#), [Row](#), [TextOrientation](#)
properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ListCount Property

Applies To

fpCombo, fpList controls

Description

Sets or returns the number of items in the list of an fpCombo or fpList control.

Syntax

C **UINT** LC_GetListCount(**HWND** hWnd, **long FAR ***lpValue);
UINT LC_SetListCount(**HWND** hWnd, **long** value);

C++ **long** **Class**::GetListCount(**void**);
Class::SetListCount(**long** value);

Visual Basic [form.]control.ListCount[= value&]

Designer Page

[Add Data designer page](#)

Remarks

The ListCount property provides the number of items in the list of an fpCombo or fpList control.

Data Type

Integer (Long)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ListDown Property

[See Also](#) [Example](#)

Applies To

fpCombo control

Description

Sets or returns whether the drop-down list is currently displayed in a drop-down fpCombo control. This property is available at run time only.

Syntax

C	UINT CbxGetListDown(HWND hWnd, BOOL FAR *pValue); UINT CbxSetListDown(HWND hWnd, BOOL value);
C++	BOOL CfpComboBox::GetListDown(void); CfpComboBox::SetListDown(BOOL value);
Visual Basic	[form.]fpCombo1.ListDown[= boolean%]

Remarks

You can use this property to determine if the drop-down list is up or down, or to force the drop-down list to be displayed or to be removed from the screen.

When the ListDown property is set to True, the drop-down fpCombo control displays its drop-down list.

You must move the focus to the fpCombo control (using the Form's Activate event or the SetFocus method) before setting the ListDown property.

Data Type

Integer (Boolean)

[Print](#)

[Copy](#)

[Close](#)

The following example sets the drop-down list to be displayed whenever the fpCombo control receives the focus. The drop-down list is automatically removed whenever the fpCombo control loses the focus.

C

```
void OnGotFocus(UNIT, int, Cwnd*, LPVOID)
{
    CbxSetListDown(hWnd, TRUE);
}
```

C++

```
void CLBDlg::OnGotFocus(UNIT, int, Cwnd*, LPVOID)
{
    m_fpCombo->SetListDown(TRUE);
}
```

Visual Basic

```
Sub fpCombo1_GotFocus ()
' Display list when fpCombo control receives focus
fpCombo1.ListDown = True
End Sub
```


See Also

[MaxDrop](#), [Style](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ListGrayAreaColor Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the color of the gray area between cells and the control border.

Syntax

C `UINT LC_GetListGrayAreaColor(HWND hWnd, COLORREF FAR *lpValue);`
`UINT LC_SetListGrayAreaColor(HWND hWnd, COLORREF value);`

C++ `COLORREF Class::GetListGrayAreaColor(void);`
`Class::SetListGrayAreaColor(COLORREF value);`

Visual Basic `[form.]control.ListGrayAreaColor[= color]`

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

The default value for the ListGrayAreaColor property is &H8000000B& (the default color, which is light gray).

Data Type

Color

See Also

[Changing the List Gray Area Color](#)

[BorderGrayAreaColor](#), [GrayAreaColor](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ListIndex Property

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the index of the currently selected row in an fpCombo or fpList control. This property is available at run time only.

Syntax

C **UINT** LC_GetListIndex(**HWND** hWnd, **long FAR ***lpValue);
UINT LC_SetListIndex(**HWND** hWnd, **long** value);

C++ **long** **Class**::GetListIndex(**void**);
Class::SetListIndex(**long** value);

Visual Basic [form.]control.ListIndex[= value&]

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The ListIndex property returns 1 when no rows are selected.

Data Type

Integer (Long)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ListLeftOffset Property

[See Also](#)

Applies To

fpCombo control

Description

Sets or returns the size in pixels of the offset of the list from the left side of an fpCombo control.

Syntax

C **UINT** CbxGetListLeftOffset(*HWND* hWnd, **long FAR** *lpValue);
UINT CbxSetListLeftOffset(*HWND* hWnd, **long** value);

C++ **long** CfpComboBox::GetListLeftOffset(**void**);
CfpComboBox::SetListLeftOffset(**long** value);

Visual Basic [form.]fpCombo1.ListLeftOffset[= value!]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the ListLeftOffset property is 1, which sets the offset to the following values:

fpCombo Style	Value
Simple combo	7 pixels
Drop-down combo	7 pixels
Drop-down list	0 pixels

Data Type

Single

See Also

[Choosing the fpCombo Control Style](#)

[Style](#) property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ListWidth Property

[See Also](#) [Example](#)

Applies To

fpCombo control

Description

Sets or returns the width of the list portion in an fpCombo control.

Syntax

C **UINT** CbxGetListWidth(**HWND** hWnd, **long FAR** *lpValue);
 UINT CbxSetListWidth(**HWND** hWnd, **long** value);

C++ **long** CfpComboBox::GetListWidth(**void**);
 CfpComboBox::SetListWidth(**long** value);

Visual Basic [form.]fpCombo1.ListWidth[= value%]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the ListWidth property is

 1, which sets the width of the list to be the same as the width of the edit field, less the width of the [ListLeftOffset](#) property setting.

In Visual Basic, the measurement unit used by the ListWidth property depends on the setting of the form's ScaleMode property. The default ScaleMode setting is twips (1/1440 of an inch). Generally the ActiveX and VBX controls use twips as the default measurement unit, and the DLL control uses pixels as the default measurement unit.

The ListWidth property does not apply when the [Style](#) property is set to 1 (Simple Combo).

When the ListWidth property setting is less than the sum of the column widths set for display, the far right columns are truncated. Use a horizontal scroll bar when the ListWidth property setting is less than the sum of the column widths.

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example creates a drop-down fpCombo control that displays a list of cities. The list width is set to accommodate the cities listed, and horizontal and vertical scroll bars are disabled unless they are needed.

C

```
LC_SetRow(hWnd, -1);
LC_SetInsertRow(hWnd, "Abilene");
LC_SetInsertRow(hWnd, "Tifton");
LC_SetInsertRow(hWnd, "Alamosa");
LC_SetInsertRow(hWnd, "Garden City");
LC_SetInsertRow(hWnd, "Glens Falls");
LC_SetInsertRow(hWnd, "Adrian");
/* Set width of list */
LC_SetListWidth(hWnd, 250);
/* Show scroll bars as disabled when not needed */
LC_SetScrollBarH(hWnd, LC_SCROLLBARH_SHOW_DISABLED);
LC_SetScrollBarV(hWnd, LC_SCROLLBARV_SHOW_DISABLED);
```

C++

```
fpCombo1->SetRow(-1);
fpCombo1->SetInsertRow("Abilene");
fpCombo1->SetInsertRow("Tifton");
fpCombo1->SetInsertRow("Alamosa");
fpCombo1->SetInsertRow("Garden City");
fpCombo1->SetInsertRow("Glens Falls");
fpCombo1->SetInsertRow("Adrian");
// Set width of list
fpCombo1->SetListWidth(250);
// Show scroll bars as disabled when not needed
fpCombo1->SetScrollBarH(LC_SCROLLBARH_SHOW_DISABLED);
fpCombo1->SetScrollBarV(LC_SCROLLBARV_SHOW_DISABLED);
```

Visual Basic

```
fpCombo1.Row = -1
fpCombo1.InsertRow = "Abilene"
fpCombo1.InsertRow = "Tifton"
fpCombo1.InsertRow = "Alamosa"
fpCombo1.InsertRow = "Garden City"
fpCombo1.InsertRow = "Glens Falls"
fpCombo1.InsertRow = "Adrian"
' Set width of list
fpCombo1.ListWidth = 2500
' Show scroll bars as disabled when not needed
fpCombo1.ScrollBarH = LC_SCROLLBARH_SHOW_DISABLED
fpCombo1.ScrollBarV = LC_SCROLLBARV_SHOW_DISABLED
```


See Also

[ColumnWidthScale](#), [ListLeftOffset](#), [Style](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

MaxDrop Property

[See Also](#)

[Example](#)

Applies To

fpCombo control

Description

Sets or returns the number of rows to display in the drop-down list of an fpCombo control.

Syntax

C **UINT** CbxGetMaxDrop(**HWND** hWnd, **short FAR** *lpValue);
 UINT CbxSetMaxDrop(**HWND** hWnd, **short** value);

C++ **short** CfpComboBox::GetMaxDrop(**void**);
 CfpComboBox::SetMaxDrop(**short** value);

Visual Basic [form.]fpCombo1.MaxDrop[= value%]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

The default value for the MaxDrop property is 8 rows.

The edit field, header, and horizontal scroll bar, when displayed, are not counted as rows.

The MaxDrop property does not apply when the [Style](#) property is set to 1 (Simple Combo).

Data Type

Integer

See Also

[Specifying the Number of Rows Displayed in the Drop-Down List](#)

[ListDown](#), [Style](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

MaxEditLen Property

[See Also](#) [Example](#)

Applies To

fpCombo control

Description

Sets or returns the maximum number of characters of text that the user can enter in the edit field.

Syntax

C	UINT CbxGetMaxEditLen(HWND hWnd, short FAR *lpValue); UINT CbxSetMaxEditLen(HWND hWnd, short value);
C++	short CfpComboBox::GetMaxEditLen(void); CfpComboBox::SetMaxEditLen(short value);
Visual Basic	[form.]fpCombo1.MaxEditLen[= value%]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

The default value for the MaxEditLen property is 150 characters.

Use the [ColumnEdit](#) property to specify which column appears in the edit field.

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example creates a five-column, simple combo style combo box control. The maximum number of characters allowed in the edit field is 10.

C

```
LC_SetColumns(hWnd, 5);  
LC_SetStyle(hWnd, CBX_STYLE_SIMPLE_COMBO);  
LC_SetMaxEditLen(hWnd, 10);
```

C++

```
fpCombo1->SetColumns(5);  
fpCombo1->SetStyle(CBX_STYLE_SIMPLE_COMBO);  
fpCombo1->SetMaxEditLen(10);
```

Visual Basic

```
fpCombo1.Columns = 5  
fpCombo1.Style = CBX_STYLE_SIMPLE_COMBO  
fpCombo1.MaxEditLen = 10
```

See Also

[Choosing the fpCombo Control Style](#)

[ColumnEdit](#) property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

MergeAdjustView Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether the view of cell contents of merged rows or columns is adjusted as the user scrolls through the list.

Syntax

C **UINT** LC_GetMergeAdjustView(**HWND** hWnd, **BOOL FAR** *lpValue);
UINT LC_SetMergeAdjustView(**HWND** hWnd, **BOOL** value);

C++ **BOOL** **Class**::GetMergeAdjustView(**void**);
Class::SetMergeAdjustView(**BOOL** value);

Visual Basic [form.]control.MergeAdjustView[= boolean%]

Designer Page

[Merge/Join designer page](#)

Remarks

The default value for the MergeAdjustView property is False. If you set the MergeAdjustView property to True, the cell contents of merged rows or columns are always vertically (in the case of merged rows) or horizontally (in the case of merged columns) centered in the portion of the cell that is displayed. The cell contents do not scroll out of the viewing area until the entire merged area is out of the viewing area.

Data Type

Integer (Boolean)

See Also

[Merging Columns or Rows](#)

[ColMerge](#), [RowMerge](#) properties

PD	RD	WR	RT	DT
	✓		✓	

MouseOverArea Property

[See Also](#) [Example](#)

Applies To

fpCombo control

Description

Returns where the mouse is positioned in an fpCombo control.

Syntax

C **UINT** CbxGetMouseOverArea(**HWND** hWnd, **short FAR** *pValue);
C++ **short** CfpComboBox::GetMouseOverArea(**void**);
Visual Basic [form.]fpCombo1.MouseOverArea

Remarks

The following values can be returned:

Return value	Description	Constant
0 - None	Mouse is not over the drop-down arrow, edit field, or list portion of the control	CBX_MOUSEOVERAREA_NONE
1 - Button	Mouse is positioned over the drop-down arrow button	CBX_MOUSEOVERAREA_BUTTON
2 - Edit	Mouse is positioned over the edit field	CBX_MOUSEOVERAREA_EDIT
3 - List	Mouse is positioned over the list portion of the control	CBX_MOUSEOVERAREA_LIST

You can use this property in events such as MouseMove, Click, or DbClick.

Data Type

Integer (Boolean)

[Print](#)

[Copy](#)

[Close](#)

The following example prints a message describing where the mouse is positioned when the user moves the mouse.

C

```
void OnButtonDblClickLB(UINT, int, Cwnd*, LPVOID)
{
    short x;

    sprintf(buffer, "%d", CbxGetMouseOverArea(hWnd, &x));
    MessageBox(buffer, "MouseOverCol", MB_OK);
}
```

C++

```
void CAboutDlg::OnButtonDblClickLB(UINT, int, Cwnd*, LPVOID)
{
    fpCombo1=(CfpComboBox*)GetDlgItem(IDC_COMBO);

    sprintf(buffer, "%d", fpCombo1->GetMouseOverArea( ));
    MessageBox(buffer, "MouseOverCol", MB_OK);
}
```

Visual Basic

```
Sub fpCombo1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
' Display what part of the combo box the mouse is over
If fpCombo1.MouseOverArea = 0 Then
    Debug.Print "Mouse is not over the arrow, edit field, or list"
ElseIf fpCombo1.MouseOverArea = 1 Then
    Debug.Print "Mouse is over the drop-down arrow"
ElseIf fpCombo1.MouseOverArea = 2 Then
    Debug.Print "Mouse is over the edit field"
Else
    Debug.Print "Mouse is over the list"
End If
End Sub
```

See Also

[MouseOverCol](#), [MouseOverColHeader](#), [MouseOverGrp](#), [MouseOverGrpHeader](#), [MouseOverRow](#) properties

PD	RD	WR	RT	DT
	✓		✓	

MouseOverCol Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Returns the number of the column over which the mouse is currently positioned in a multiple-column fpCombo or fpList control. This property is available at run time only.

Syntax

C **UINT** LC_GetMouseOverCol(*HWND* hWnd, *short FAR* *pValue);

C++ **short Class::**GetMouseOverCol(**void**);

Visual Basic [*form.*]control.MouseOverCol

Remarks

This property returns the number of the column over which the mouse pointer is currently positioned. Column numbers start with zero, which specifies the top, leftmost column.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The MouseOverCol property returns 1 when the cursor is not positioned on the fpCombo or fpList control or on a specific column. The MouseOverCol property returns

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

1 for single-column fpCombo or fpList controls (when the [Columns](#) property is set to 0).

Use the [MouseOverColHeader](#) property to distinguish between the header and the body of the column.

Data Type

Integer

See Also

[Columns](#), [MouseOverArea](#), [MouseOverColHeader](#), [MouseOverGrp](#), [MouseOverGrpHeader](#), [MouseOverRow](#) properties

PD	RD	WR	RT	DT
	✓		✓	

MouseOverColHeader Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Returns the number of the column header over which the mouse is currently positioned in a multiple-column fpCombo or fpList control. This property is available at run time only.

Syntax

C **UINT** LC_GetMouseOverColHeader(*HWND hWnd*, **long FAR** *pValue);

C++ **long** *Class*::GetMouseOverColHeader(**void**);

Visual Basic [*form.*]control.MouseOverColHeader

Remarks

This property returns the number of the column header over which the mouse pointer is currently positioned. Column numbers start with zero, which specifies the top, leftmost column header.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The MouseOverColHeader property returns 1 when the cursor is not positioned on the fpCombo or fpList control or on a specific column header. The MouseOverColHeader property returns

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

1 for single-column fpCombo or fpList controls (when the [Columns](#) property is set to 0).

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example creates a three-column fpList control with customized column header text. To try this example, create an fpList control and a Text control on your form. When you run the example, as you move the mouse over the fpList control's column headers, the Text control displays the column header text.

Visual Basic

```
Sub Form_Load ()
    fpList1.Columns = 3
    fpList1.Row = -1
    fpList1.InsertRow = "ACME" & Chr$(9) & "4459" & Chr$(9) & "(919) 555-3275"
    fpList1.InsertRow = "Buster" & Chr$(9) & "4528" & Chr$(9) & "(919) 555-1982"
    fpList1.InsertRow = "Caref" & Chr$(9) & "4656" & Chr$(9) & "(919) 555-0378"
    fpList1.ColumnHeaderShow = True
    fpList1.Col = 0
    fpList1.ColHeaderText = "Publisher"
    fpList1.Col = 1
    fpList1.ColHeaderText = "ID Number"
    fpList1.Col = 2
    fpList1.ColHeaderText = "Fax Number"
End Sub

Sub fpList1_MouseMove (Button As Integer, Shift As Integer, x As Single, y As Single)
    k = fpList1.MouseOverColHeader
    If k >= 0 Then
        fpList1.Col = k
        Text1.Text = fpList1.ColHeaderText
    Else
        Text1.Text = ""
    End If
End Sub
```

See Also

[Columns](#), [MouseOverArea](#), [MouseOverCol](#), [MouseOverGrp](#), [MouseOverGrpHeader](#), [MouseOverRow](#) properties

PD	RD	WR	RT	DT
	✓		✓	

MouseOverGrp Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Returns the number of the group over which the mouse is currently positioned in a multiple-group fpCombo or fpList control. This property is available at run time only.

Syntax

C **UINT** LC_GetMouseOverGrp(*HWND hWnd*, *short FAR *pValue*);

C++ **short Class::**GetMouseOverGrp(*void*);

Visual Basic [*form.*]control.MouseOverGrp

Remarks

This property returns the number of the group over which the mouse pointer is currently positioned. Group numbers start with zero, which specifies the top, leftmost group.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The MouseOverGrp property returns 1 when the cursor is not positioned on the fpCombo or fpList control or on a specific group.

Use the [MouseOverGrpHeader](#) property to distinguish between the header and the body of the group.

Data Type

Integer

See Also

[Groups](#), [MouseOverArea](#), [MouseOverCol](#), [MouseOverColHeader](#), [MouseOverGrpHeader](#), [MouseOverRow](#) properties

PD	RD	WR	RT	DT
	✓		✓	

MouseOverGrpHeader Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Returns the number of the group header over which the mouse is currently positioned in a multiple-group fpCombo or fpList control. This property is available at run time only.

Syntax

C **UINT** LC_GetMouseOverGrpHeader(**HWND** hWnd, **short FAR** *pValue);

C++ **short Class::**GetMouseOverGrpHeader(**void**);

Visual Basic [form.]control.MouseOverGrpHeader

Remarks

This property tells you the group header the user is pointing to at run time. Group numbers start with zero, which specifies the top, leftmost group header.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The MouseOverGrpHeader property returns 1 when the cursor is not positioned on the fpCombo or fpList control or on a specific group header.

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example creates a three-group fpList control with customized group header text. To try this example, create an fpList control and a Text control on your form. When you run the example, as you move the mouse over the fpList control's group headers, the Text control displays the group header text.

Visual Basic

```
Sub Form_Load ()
    fpList1.Groups = 3
    fpList1.Columns = 3
    fpList1.ColumnHeaderShow = False
    fpList1.Row = -1
    fpList1.InsertRow = "ACME" & Chr$(9) & "4459" & Chr$(9) & "(919) 555-3275"
    fpList1.InsertRow = "Buster" & Chr$(9) & "4528" & Chr$(9) & "(919) 555-1982"
    fpList1.InsertRow = "Caref" & Chr$(9) & "4656" & Chr$(9) & "(919) 555-0378"
    fpList1.GroupHeaderShow = True
    fpList1.Grp = 0
    fpList1.GrpHeaderText = "Publisher"
    fpList1.Grp = 1
    fpList1.GrpHeaderText = "ID Number"
    fpList1.Grp = 2
    fpList1.GrpHeaderText = "Fax Number"
    fpList1.Col = 0
    fpList1.ColParentGroup = 0
    fpList1.Col = 1
    fpList1.ColParentGroup = 1
    fpList1.Col = 2
    fpList1.ColParentGroup = 2
End Sub

Sub fpList1_MouseMove (Button As Integer, Shift As Integer, x As Single, y As Single)
    k = fpList1.MouseOverGrpHeader
    If k >= 0 Then
        fpList1.Grp = k
        Text1.Text = fpList1.GrpHeaderText
    Else
        Text1.Text = ""
    End If
End Sub
```

See Also

[Groups](#), [MouseOverArea](#), [MouseOverCol](#), [MouseOverColHeader](#), [MouseOverGrp](#) properties

PD	RD	WR	RT	DT
	✓		✓	

MouseOverRow Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Returns the number of the row over which the mouse is currently positioned in an fpCombo or fpList control. This property is available at run time only.

Syntax

C **UINT** LC_GetMouseOverRow(*HWND hWnd*, *long FAR *pValue*);

C++ **long Class::**GetMouseOverRow(**void**);

Visual Basic [*form.*]control.MouseOverRow

Remarks

This property returns the number of the row (list item) over which the mouse pointer is currently positioned. Row numbers start with zero, which specifies the first row.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The MouseOverRow property returns a value of 1 when the cursor is not positioned on the fpCombo or fpList control or on a row. The MouseOverRow property returns the same value for single- and multiple-column lists.

Data Type

Integer

See Also

[Columns](#), [MouseOverArea](#), [MouseOverCol](#), [MouseOverColHeader](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

MultiLine Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether text displays on a single line or on multiple lines.

Syntax

C *UINT* LC_GetMultiLine(*HWND* hWnd, *short FAR *pValue*);
UINT LC_SetMultiLine(*HWND* hWnd, *short value*);

C++ *short Class::*GetMultiLine(*void*);
*Class::*SetMultiLine(*short value*);

Visual Basic [form.]control.MultiLine[= *setting%*]

Designer Page

[List subtab of the ApplyTo designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Default	Uses the same setting as the hierarchical predecessor	LC_MULTILINE_DEFAULT
1 - Single Line	(Default) Text displays on one line	LC_MULTILINE_SINGLE_LINE
2 - Multiple Line	Text displays on multiple lines	LC_MULTILINE_MULTIPLE_LINE

Use the [ListApplyTo](#) property to specify the area of the list to which the MultiLine property applies.

You must use the [RowHeight](#) property to increase the height of each row that displays text on multiple lines.

Data Type

Integer (Enumerated)

See Also

[Wrapping Text in a List Pro Control](#)

[ListApplyTo](#), [RowHeight](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

MultiSelect Property

[See Also](#)

[Example](#)

Applies To

fpList control

Description

Sets or returns whether and when the user can select multiple items in an fpList control.

Syntax

C **UINT** LbxGetMultiSelect(**HWND** hWnd, **short FAR *lpValue**);
UINT LbxSetMultiSelect(**HWND** hWnd, **short value**);

C++ **short** CfpListBox::GetMultiSelect(**void**);
CfpListBox::SetMultiSelect(**short value**);

Visual Basic [form.]fpList1.MultiSelect[= setting%]

Designer Page

[Miscellaneous designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - None	(Default) Prevents multiple selections	LBX_MULTISELECT_NONE
1 - Simple	Allows user to click the mouse button or to press the Spacebar to select or deselect an item in the list and to use the arrow keys to move the focus	LBX_MULTISELECT_SIMPLE
2 - Extended	Allows user to either	LBX_MULTISELECT_EXTENDED

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Press Shift and click or press Shift and an arrow key to extend the highlighting to the current item

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Press Ctrl and click to select or deselect items

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates an fpList control that lets the user select all the items in the list by double-clicking the control.

Visual Basic

```
Sub Form_Load ()
fpList1.Row = -1
fpList1.InsertRow = "North Carolina"
fpList1.InsertRow = "Ohio"
fpList1.InsertRow = "Virginia"
fpList1.InsertRow = "Florida"
fpList1.InsertRow = "California"
fpList1.MultiSelect = LBX_MULTISELECT_EXTENDED
End Sub
```

```
Sub fpList1_DblClick ()
fpList1.Action = LC_ACTION_SELECTALL
End Sub
```

See Also

[Accessing List Items](#)

[Action](#), [NextSel](#) properties

PD	RD	WR	RT	DT
	✓		✓	

NewIndex Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Returns the index of the item most recently added to an fpCombo or fpList control. This property is available at run time only.

Syntax

C **UINT** LC_GetNewIndex(*HWND* hWnd, **long FAR** *lpValue);

C++ **long Class::**GetNewIndex(**void**);

Visual Basic [form.]control.NewIndex

Remarks

This property is useful with sorted lists when you need a list of values that correspond to each item in the ItemData property array. As you add an item in a sorted list, the item is inserted in the list alphabetically. This property tells you where the item was inserted so you can insert a corresponding value using the [ItemData](#) property at the same index.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The NewIndex property returns 1 if there are no items in the list or if an item has been deleted since the last item was added.

Data Type

Integer (Long)

See Also

[ItemData](#) property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

NextSel Property

[See Also](#) [Example](#)

Applies To

fpList control

Description

Sets or returns the number of the next selected row when multiple rows are selected. This property is available at run time only.

Syntax

C	UINT LbxGetNextSel(HWND hWnd, long FAR *pValue); UINT LbxSetNextSel(HWND hWnd, long value);
C++	long CfpListBox::GetNextSel(void); CfpListBox::SetNextSel(long value);
Visual Basic	[form.]fpList1.NextSel[= value&]

Remarks

Use the [SelCount](#) property to determine how many rows have been selected.

Data Type

Integer (Long)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a list box control that allows simple multiple selection of list items. When the user selects an item in a list, the list item is displayed in a text box.

Visual Basic

```
Sub Form_Load()  
    m_LB.MultiSelect = LC_MULTISELECT_SIMPLE  
End Sub  
  
Sub m_LB_SelChange(ItemIndex As Long)  
    Text1.Text = m_LB.NextSel()  
End Sub
```


See Also

[MultiSelect](#), [SelCount](#), [Selected](#), [SelMax](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

NoIntegralHeight Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether the list of an fpCombo or fpList control is automatically resized so that an entire row is displayed on the bottom.

Syntax

C `UINT LC_GetNoIntegralHeight(HWND hWnd, BOOL FAR *pValue);`
`UINT LC_SetNoIntegralHeight(HWND hWnd, BOOL value);`

C++ `BOOL Class::GetNoIntegralHeight(void);`
`Class::SetNoIntegralHeight(BOOL value);`

Visual Basic `[form.]control.NoIntegralHeight[= boolean%]`

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

The default value for this property is False, which resizes the list when necessary to display an entire row at the bottom.

When the value of this property is True, the list does not automatically resize.

Data Type

Integer (Boolean)

See Also

[RowHeight](#) property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Picture Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the graphic that is displayed in a list item. This property is available at run time only.

Syntax

C **UINT** LC_GetPicture(**HWND** hWnd, **short FAR** *retPictType,
↳ **short FAR** *retPictHndType, **LPVOID** buffer,

↳ **UINT** nBufferSize, **HPALETTE FAR** *retHPal);

LC_SetPicture(**HWND** hWnd, **short** pictType, **short** pictHndType,

↳ **DWORD** pictHndOrRes, **HPALETTE** hPal);

C++ **LPVOID** **Class**::GetPicture(**short FAR*** retPictType,
↳ **short FAR*** retPictHndType, **LPVOID** buffer,

↳ **UINT** bufferSize, **HPALETTE FAR*** retHPal);

Class::SetPicture(**short** retPictType, **short** retHndType,

↳ **DWORD** buffer, **HPALETTE** hPal);

Visual Basic [form.]control.Picture[= picture]

Designer Page

[List subtab of the ApplyTo designer page](#)

Remarks

You can display small graphics (bitmaps or icons) as list items in an fpCombo or fpList control. You can display text and graphics simultaneously. The Picture property specifies the graphic displayed when a list item is not selected. In contrast, the [PictureSel](#) property specifies the graphic displayed when a list item is selected. For example, you can display a closed book icon when a list item is not selected and an open book icon when an item is selected.

You can use the [ListApplyTo](#) property to specify where the graphic is displayed.

Use the [PictureAlignH](#) and [PictureAlignV](#) properties to specify how the graphic is horizontally and vertically aligned.

Data Type

Picture

[Print](#)

[Copy](#)

[Close](#)

The following example sets the row height to the height of an icon displayed in the second row.

C

```
LC_SetColumns(hWnd, 1);
LC_SetColumnHeaderShow(hWnd, FALSE);
LC_SetInsertRow(hWnd, "Row 1");
LC_SetInsertRow(hWnd, "Row 2");
LC_SetInsertRow(hWnd, "Row 3");
/* Set RowHeight to height of an icon in twips */
LC_SetRowHeight(hWnd, Screen.TwipsPerPixelY * 32);
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SINGLE_ITEM);
LC_SetRow(hWnd, 1);
/* Load icon */
HICON hiconMail;
hbmpMail = LoadBitmap(hInstance, "mail01a");
LC_SetPicture(hWnd, FP_PICT_TYPE_ICON, FP_HNDTYPE_HANDLE, hiconMail, hPalette);
```

C++

```
fpList1->SetColumns(1);
fpList1->SetColumnHeaderShow(FALSE);
fpList1->SetInsertRow("Row 1");
fpList1->SetInsertRow("Row 2");
fpList1->SetInsertRow("Row 3");
// Set RowHeight to height of an icon in twips
fpList1->SetRowHeight(Screen.TwipsPerPixelY * 32);
fpList1->SetListApplyTo(LC_LISTAPPLYTO_SINGLE_ITEM);
fpList1->SetRow(1);
// Convert hIcon to hPic
PIC picture;
HPIC hPic;
HICON hIcon;
hIcon=LoadIcon(AfxGetInstanceHandle( ), "mail01");
picture.picData.ico.hIcon=hIcon;
picture.picType=PICTURE_ICON;
hPic=AfxSetPict(NULL,&picture);
fpList1->SetPicture(hPic);
```

Visual Basic

```
fpList1.Columns = 1
fpList1.ColumnHeaderShow = False
fpList1.InsertRow = "Row 1"
fpList1.InsertRow = "Row 2"
fpList1.InsertRow = "Row 3"
' Set RowHeight to height of an icon in twips
fpList1.RowHeight = Screen.TwipsPerPixelY * 32
fpList1.ListApplyTo = LC_LISTAPPLYTO_SINGLE_ITEM
fpList1.Row = 1
fpList1.Picture = LoadPicture("c:\listpro\icons\mail01a.ico")
```

See Also

[Displaying Graphics](#)

[ListApplyTo](#), [PictureAlignH](#), [PictureAlignV](#), [PictureSel](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

PictureAlignH Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the horizontal alignment of a bitmap or icon in a list item.

Syntax

C *UINT* LC_GetPictureAlignH(*HWND* hWnd, *short FAR* *lpValue);
UINT LC_SetPictureAlignH(*HWND* hWnd, *short* value);

C++ *short* *Class*::GetPictureAlignH(*void*);
Class::SetPictureAlignH(*short* value);

Visual Basic [form.]control.PictureAlignH[= setting%]

Designer Page

[List subtab of the ApplyTo designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Default	Uses the same setting as the hierarchical predecessor	LC_PICTUREALIGNH_DEFAULT
1 - Left	Left-justifies the graphic	LC_PICTUREALIGNH_LEFT
2 - Center	Centers the graphic	LC_PICTUREALIGNH_CENTER
3 - Right	Right-justifies the graphic	LC_PICTUREALIGNH_RIGHT
4 - Left of Text	(Default) Displays the graphic to the left of the text	LC_PICTUREALIGNH_LEFT_OF_TEXT
5 - Right of Text	Displays the graphic to the right of the text	LC_PICTUREALIGNH_RIGHT_OF_TEXT

If you set the PictureAlignH property to 4 (Left of Text) or 5 (Right of Text), the text is displayed to the right or left of the graphic, not immediately next to it.

You can use the [ListApplyTo](#) property to specify the area of the control to which to apply the horizontal alignment of graphics.

Use the [PictureAlignV](#) property to specify the vertical alignment of graphics.

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a two column list box. The first column contains a graphic and the second column contains text. The graphic is displayed at the center and top of the cell.

C

```
LC_SetColumns(hWnd, 2);
LC_SetColumnHeaderShow(hWnd, TRUE);
LC_SetLineStyle(hWnd, LC_LISTSTYLE_LOWERED);
LC_SetRowHeight(hWnd, 45);
LC_SetInsertRow(hWnd, "\t");
/* Define columns */
LC_SetCol(hWnd, 0);
LC_SetColName(hWnd, "Pix");
LC_SetColHeaderText(hWnd, "Picture");
LC_SetCol(hWnd, 1);
LC_SetColName(hWnd, "Desc");
LC_SetColHeaderText(hWnd, "Description");
/* Specify graphic and text */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SINGLE_ITEM);
LC_SetColFromName(hWnd, "Pix");
LC_SetRow(hWnd, 0);
/* Load bitmap */
HBITMAP hbmpBalloon;
hbmpBalloon = LoadBitmap(hInstance, "Balloon");
LC_SetPicture(hWnd, FP_PICT_TYPE_BITMAP, FP_HNDTYPE_HANDLE, hbmpBalloon, hPalette);
/* Add column text */
LC_SetColFromName(hWnd, "Desc");
LC_SetColText(hWnd, "Balloon");
/* Align graphic in first column */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SINGLE_ITEM);
LC_SetColFromName(hWnd, "Pix");
LC_SetPictureAlignH(hWnd, LC_PICTUREALIGNH_CENTER);
LC_SetPictureAlignV(hWnd, LC_PICTUREALIGNV_TOP);
```

C++

```
fpList1->SetColumns(2);
fpList1->SetColumnHeaderShow(TRUE);
fpList1->SetLineStyle(LC_LISTSTYLE_LOWERED);
fpList1->SetRowHeight(45);
fpList1->SetInsertRow("\t");
// Define columns
fpList1->SetCol(0);
fpList1->SetColName("Pix");
fpList1->SetColHeaderText("Picture");
fpList1->SetCol(1);
fpList1->SetColName("Desc");
fpList1->SetColHeaderText("Description");
// Specify graphic and text
fpList1->SetListApplyTo(LC_LISTAPPLYTO_SINGLE_ITEM);
fpList1->SetColFromName("Pix");
fpList1->SetRow(0);
// Convert hBitmap to hPic
PIC picture;
HPIC hPic;
HBITMAP hBitmap;
hBitmap=LoadBitmap(AfxGetInstanceHandle( ), "balloon");
picture.picData.bmp.hbitmap=hbitmap;
picture.picType=PICTYPE_BITMAP;
hPic=AfxSetPict(NULL, &picture);
fpList1->SetPicture(hPic);
// Add column text
fpList1->SetColFromName("Desc");
fpList1->SetColText("Balloon");
```



```
// Align graphic in first column
fpList1->SetListApplyTo(LC_LISTAPPLYTO_SINGLE_ITEM);
fpList1->SetColFromName("Pix");
fpList1->SetPictureAlignH(LC_PICTUREALIGNH_CENTER);
fpList1->SetPictureAlignV(LC_PICTUREALIGNV_TOP);
```

Visual Basic

```
fpList1.Columns = 2
fpList1.ColumnHeaderShow = True
fpList1.LineStyle = LC_LISTSTYLE_LOWERED
fpList1.RowHeight = 600
fpList1.InsertRow = "" & Chr$(9) & ""
' Define columns
fpList1.Col = 0
fpList1.ColName = "Pix"
fpList1.ColHeaderText = "Picture"
fpList1.Col = 1
fpList1.ColName = "Desc"
fpList1.ColHeaderText = "Description"
' Specify graphic and text
fpList1.ListApplyTo = LC_LISTAPPLYTO_SINGLE_ITEM
fpList1.ColFromName = "Pix"
fpList1.Row = 0
fpList1.Picture = LoadPicture("c:\vb16\bitmaps\assorted\balloon.bmp")
' Add column text
fpList1.ColFromName = "Desc"
fpList1.ColText = "Balloon"
' Align graphic in first column
fpList1.ListApplyTo = LC_LISTAPPLYTO_SINGLE_ITEM
fpList1.ColFromName = "Pix"
fpList1.PictureAlignH = LC_PICTUREALIGNH_CENTER
fpList1.PictureAlignV = LC_PICTUREALIGNV_TOP
```

See Also

[Aligning Text and Graphics](#)

[ListApplyTo](#), [Picture](#), [PictureAlignV](#), [PictureSel](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

PictureAlignV Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the vertical alignment of a bitmap or icon in a list item.

Syntax

C *UINT* LC_GetPictureAlignV(*HWND* hWnd, *short FAR* *lpValue);
UINT LC_SetPictureAlignV(*HWND* hWnd, *short* value);

C++ *short Class::*GetPictureAlignV(*void*);
*Class::*SetPictureAlignV(*short* value);

Visual Basic [form.]control.PictureAlignV[= setting%]

Designer Page

[List subtab of the ApplyTo designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Default	Uses the same setting as the hierarchical predecessor	LC_PICTUREALIGNV_DEFAULT
1 - Top	Displays the graphic at the top of the row	LC_PICTUREALIGNV_TOP
2 - Center	(Default) Displays the graphic in the center of the row	LC_PICTUREALIGNV_CENTER
3 - Bottom	Displays the graphic at the bottom of the row	LC_PICTUREALIGNV_BOTTOM
4 - Top of Text	Displays the graphic above the text	LC_PICTUREALIGNV_TOP_OF_TEXT
5 - Bottom of Text	Displays the graphic below the text	LC_PICTUREALIGNV_BOTTOM_OF_TEXT

You can use the [ListApplyTo](#) property to specify the area of the control to which to apply the vertical alignment of graphics.

Use the [PictureAlignH](#) property to specify the horizontal alignment of graphics.

Data Type

Integer (Enumerated)

See Also

[Aligning Text and Graphics](#)

[ListApplyTo](#), [Picture](#), [PictureAlignH](#), [PictureSel](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

PictureSel Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the graphic to display when a row is selected. This property is available at run time only.

Syntax

C **UINT** LC_GetPictureSel(**HWND** hWnd, **short FAR** *retPictType,
↳ **short FAR** *retPictHndType, **LPVOID** buffer,

↳ **UINT** nBufferSize, **HPALETTE FAR** *retHPal);

LC_SetPictureSel(**HWND** hWnd, **short** pictType, **short** pictHndType,

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

DWORD pictHndOrRes, **HPALETTE** hPal);

C++ **LPVOID** [Class](#)::GetPictureSel(**short FAR*** retPictType,

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

short FAR* retPictHndType, **LPVOID** buffer,

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

UINT bufferSize, **HPALETTE FAR*** retHPal);

[Class](#)::SetPictureSel(**short** retPictType, **short** retHndType,

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

DWORD buffer, **HPALETTE** hPal);

Visual Basic [form.]control.PictureSel[= picture]

Designer Page

[List subtab of the ApplyTo designer page](#)

Remarks

You can display small graphics (bitmaps or icons) as list items in an fpCombo or fpList control. You can display text and graphics simultaneously. The PictureSel property specifies the graphic displayed when a list item is selected. In contrast, the [Picture](#) property specifies the graphic displayed when a list item is not selected. For example, you can display a closed book icon when a list item is not selected and an open book icon when an item is selected.

If you want to specify that the graphic appear in a particular column when a row is selected, set the [Col](#), [ColFromID](#), or [ColFromName](#) property before setting the PictureSel property.

You can use the [ListApplyTo](#) property to specify where the graphic is displayed.

Use the [PictureAlignH](#) and [PictureAlignV](#) properties to specify how the graphic is horizontally and vertically aligned.

Data Type

Picture

[Print](#)

[Copy](#)

[Close](#)

The following example creates a list box control with two columns. When the first row is selected, a balloon icon is displayed. When the second row is selected, a calendar icon is displayed.

C

```
LC_SetColumns(hWnd, 2);
LC_SetColumnHeaderShow(hWnd, TRUE);
LC_SetLineStyle(hWnd, LC_LISTSTYLE_LOWERED);
LC_SetRowHeight(hWnd, 45);
LC_SetInsertRow(hWnd, "\tBalloon");
LC_SetInsertRow(hWnd, "\tCalendar");
/* Define columns */
LC_SetCol(hWnd, 0);
LC_SetColName(hWnd, "Pix");
LC_SetColHeaderText(hWnd, "Picture");
LC_SetCol(hWnd, 1);
LC_SetColName(hWnd, "Desc");
LC_SetColHeaderText(hWnd, "Description");
/* Graphic to display when first row is selected */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SINGLE_ITEM);
LC_SetColFromName(hWnd, "Pix");
LC_SetRow(hWnd, 0);
/* Load bitmap */
HBITMAP hbmpBalloon;
hbmpBalloon = LoadBitmap(hInstance, "balloon");
LC_SetPicture(hWnd, FP_PICT_TYPE_BITMAP, FP_HNDTYPE_HANDLE, hbmpBalloon, hPalette);
/* Graphic to display when second row is selected */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SINGLE_ITEM);
LC_SetColFromName(hWnd, "Pix");
LC_SetRow(hWnd, 1);
HBITMAP hbmpCalendar;
hbmpCalendar = LoadBitmap(hInstance, "calendar");
LC_SetPicture(hWnd, FP_PICT_TYPE_BITMAP, FP_HNDTYPE_HANDLE, hbmpCalendar, hPalette);
```

C++

```
fpList1->SetColumns(2);
fpList1->SetColumnHeaderShow(TRUE);
fpList1->SetLineStyle(LC_LISTSTYLE_LOWERED);
fpList1->SetRowHeight(45);
fpList1->SetInsertRow("\tBalloon");
fpList1->SetInsertRow("\tCalendar");
// Define columns
fpList1->SetCol(0);
fpList1->SetColName("Pix");
fpList1->SetColHeaderText("Picture");
fpList1->SetCol(1);
fpList1->SetColName("Desc");
fpList1->SetColHeaderText("Description");
// Graphic to display when first row is selected
fpList1->SetListApplyTo(LC_LISTAPPLYTO_SINGLE_ITEM);
fpList1->SetColFromName("Pix");
fpList1->SetRow(0);
// Convert hBitmap to hPic
PIC picture;
HPIC hPic;
HBITMAP hBitmap;
hBitmap=LoadBitmap(AfxGetInstanceHandle( ), "balloon");
picture.picData.bmp.hbitmap=hbitmap;
picture.picType=PICTYPE_BITMAP;
hPic=AfxSetPict(NULL, &picture);
fpList1->SetPicture(hPic);
// Graphic to display when second row is selected
fpList1->SetListApplyTo(LC_LISTAPPLYTO_SINGLE_ITEM);
```

```

fpList1->SetColFromName("Pix");
fpList1->SetRow(1);
// Convert hBitmap to hPic
PIC picture;
HPIC hPic;
HBITMAP hBitmap;
hBitmap=LoadBitmap(AfxGetInstanceHandle( ), "calendar");
picture.picData.bmp.hbitmap=hbitmap;
picture.picType=PICTYPE_BITMAP;
hPic=AfxSetPict(NULL, &picture);
fpList1->SetPicture(hPic);

```

Visual Basic

```

fpList1.Columns = 2
fpList1.ColumnHeaderShow = True
fpList1.LineStyle = 3 'LC_LINESTYLE_LOWERED
fpList1.RowHeight = 600
fpList1.InsertRow = "" & Chr$(9) & "Balloon"
fpList1.InsertRow = "" & Chr$(9) & "Calendar"
' Define columns
fpList1.Col = 0
fpList1.ColName = "Pix"
fpList1.ColHeaderText = "Picture"
fpList1.Col = 1
fpList1.ColName = "Desc"
fpList1.ColHeaderText = "Description"
' Graphic to display when first row is selected
fpList1.ListApplyTo = LC_LISTAPPLYTO_SINGLE_ITEM
fpList1.ColFromName = "Pix"
fpList1.Row = 0
fpList1.PictureSel = LoadPicture("c:\vb16\bitmaps\assorted\balloon.bmp")
' Graphic to display when second row is selected
fpList1.ListApplyTo = LC_LISTAPPLYTO_SINGLE_ITEM
fpList1.ColFromName = "Pix"
fpList1.Row = 1
fpList1.PictureSel = LoadPicture("c:\vb16\bitmaps\assorted\calendar.bmp")

```

See Also

[Displaying Graphics](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ListApplyTo](#), [Picture](#), [PictureAlignH](#), [PictureAlignV](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ReadOnly Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether an fpCombo or fpList control allows the user to select list items.

Syntax

C `UINT LC_GetReadOnly(HWND hWnd, BOOL FAR *pValue);`
`UINT LC_SetReadOnly(HWND hWnd, BOOL value);`

C++ `BOOL Class::GetReadOnly(void);`
`Class::SetReadOnly(BOOL value);`

Visual Basic `[form.]control.ReadOnly[= boolean%]`

Designer Page

[Miscellaneous designer page](#)

Remarks

The default value for the ReadOnly property is False, which allows the user to select list items by clicking a mouse button while the pointer is on an item or by pressing the arrow keys to scroll up and down the list.

Setting the ReadOnly property to True creates an fpCombo or fpList control that displays a list of items that cannot be selected. A read-only fpCombo or fpList control is useful for displaying multiple rows of data without letting the user highlight individual values or rows. When the ReadOnly property is set to True, the user can still scroll through the list using the scroll bar, as well as move or resize columns, as specified by the [AllowColDragDrop](#) and [AllowColResize](#) properties.

Data Type

Integer (Boolean)

See Also

[AllowColDragDrop](#), [AllowColResize](#), [Columns](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Row Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns a row in an fpCombo or fpList control. This property is available at run time only.

Syntax

C *UINT* LC_GetRow(*HWND* hWnd, *long FAR *lpValue*);
UINT LC_SetRow(*HWND* hWnd, *long value*);

C++ *longClass*::GetRow(*void*);
Class::SetRow(*long value*);

Visual Basic [form.]control.Row[= value&]

Designer Pages

Row box on:

[List subtab of the ApplyTo designer page](#)

[Line subtab of the ApplyTo designer page](#)

[Add Data designer page](#)

[Misc subtab of the Appearance designer page](#)

[Merge/Join designer page](#)

Remarks

The default value for the Row property is 0, which designates the first row of the list. To choose an entire row, set the [Col](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

property to 1 before setting the Row property.

You must specify a row by setting the Row property before retrieving data with the [List](#) or [Collist](#) properties.

Data Type

Integer (Long)

See Also

[Adding List Items](#)

[Columns, Rows, and Cells](#)

[Col](#), [ColList](#), [ExtendRow](#), [InsertRow](#), [List](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

RowHeight Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the height in twips of rows in the list portion of an fpCombo or fpList control.

Syntax

C **UINT** LC_GetRowHeight(**HWND** hWnd, **long FAR ***lpValue);
UINT LC_SetRowHeight(**HWND** hWnd, **long** value);

C++ **long** **Class**::GetRowHeight(**void**);
Class::SetRowHeight(**long** value);

Visual Basic [form.]control.RowHeight[= value%]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the RowHeight property is 1, which sets the row height based on the size of the font used in the fpCombo and fpList controls. To specify the height of a particular row, set the [Row](#) property before setting the RowHeight property.

If you have multiple levels of columns in a row, the height of every level is equal to the row height. For example, if the control has three levels and the row height is set to 100 twips, each level within the row is 100 twips high.

In Visual Basic, the measurement unit used by the RowHeight property depends on the setting of the form's ScaleMode property. The default ScaleMode setting is twips (1/1440 of an inch). Generally, the ActiveX and VBX controls use twips as the default measurement unit, and the DLL controls use pixels as the default measurement unit.

Data Type

Integer (Long)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a three-column fpList control. Text wraps in the second column. Solid horizontal lines are displayed between list items.

C

```
LC_SetColumns(hWnd, 3);
LC_SetColumnHeaderShow(FALSE);
/* Insert data */
LC_SetRow(hWnd, -1);
LC_SetInsertRow(hWnd, "AAA\t1500 Blue Ridge Road Suite 215 Raleigh, NC 27588\t(919)
555-1234");
LC_SetInsertRow(hWnd, "ABA\t20394 Capital Boulevard Raleigh, NC 27583\t(919) 555-
5678");
LC_SetInsertRow(hWnd, "ACA\t7893 Glenwood Avenue Suite 2003 Raleigh, NC 27588\t(919)
555-9012");
LC_SetLineApplyTo(hWnd, LC_LINEAPPLYTO_ROWS);
LC_SetLineStyle(hWnd, LC_LINESTYLE_SOLID);
LC_SetCol(hWnd, 0);
LC_SetColWidth(hWnd, 10);
LC_SetCol(hWnd, 1);
/* Wrap text in column */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SINGLE_ITEM);
LC_SetMultiLine(hWnd, LC_MULTILINE_MULTIPLE_LINE);
LC_SetColWidth(hWnd, 25);
LC_SetRowHeight(hWnd, 60);
```

C++

```
fpList1->SetColumns(3);
fpList1->SetColumnHeaderShow(FALSE);
// Insert data
fpList1->SetRow(-1);
fpList1->SetInsertRow("AAA\t1500 Blue Ridge Road Suite 215 Raleigh, NC 27588\t(919)
555-1234");
fpList1->SetInsertRow("ABA\t20394 Capital Boulevard Raleigh, NC 27583\t(919) 555-
5678");
fpList1->SetInsertRow("ACA\t7893 Glenwood Avenue Suite 2003 Raleigh, NC 27588\t(919)
555-9012");
fpList1->SetLineApplyTo(LC_LINEAPPLYTO_ROWS);
fpList1->SetLineStyle(LC_LINESTYLE_FLAT);
fpList1->SetCol(0);
fpList1->SetColWidth(10);
fpList1->SetCol(1);
// Wrap text in column
fpList1->SetListApplyTo(LC_LISTAPPLYTO_SINGLE_ITEM);
fpList1->SetMultiLine(LC_MULTILINE_MULTIPLE_LINE);
fpList1->SetColWidth(25);
fpList1->SetRowHeight(60);
```

Visual Basic

```
fpList1.Columns = 3
fpList1.ColumnHeaderShow = False
' Insert data
fpList1.Row = -1
fpList1.InsertRow = "AAA" & Chr$(9) & "1500 Blue Ridge Road Suite 215 Raleigh, NC
27588" & Chr$(9) & "(919) 555-1234"
fpList1.InsertRow = "ABA" & Chr$(9) & "20394 Capital Boulevard Raleigh, NC 27583" &
Chr$(9) & "(919) 555-5678"
fpList1.InsertRow = "ACA" & Chr$(9) & "7893 Glenwood Avenue Suite 2003 Raleigh, NC
27588" & Chr$(9) & "(919) 555-9012"
fpList1.LineApplyTo = LC_LINEAPPLYTO_ROWS
fpList1.LineStyle = LC_LINESTYLE_FLAT
fpList1.Col = 0
fpList1.ColWidth =10
```

```
fpList1.Col = 1
' Wrap text in column
fpList1.ListApplyTo = LC_LISTAPPLYTO_SINGLE_ITEM
fpList1.MultiLine = LC_MULTILINE_MULTIPLE_LINE
fpList1.ColWidth = 25
fpList1.RowHeight = 600
```

See Also

[Wrapping Text in a List Pro Control](#)

[EditHeight](#), [MultiLine](#), [Row](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

RowMerge Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether and how cells within a row with the same contents should be grouped in a single cell spanning multiple columns.

Syntax

C **UINT** LC_GetRowMerge(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetRowMerge(**HWND** hWnd, **short** value);

C++ **short Class::**GetRowMerge(**void**);
Class::SetRowMerge(**short** value);

Visual Basic [form.]control.RowMerge[= setting%]

Designer Page

[Merge/Join designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Off	(Default) Cells that contain the same text are not merged	LC_ROWMERGE_OFF
1 - Always	Cells that contain the same text are always merged	LC_ROWMERGE_ALWAYS
2 - Restricted	Cells that contain the same text are merged only when adjacent cells to the left and to the top are also merged	LC_ROWMERGE_RESTRICTED

You must set the [Row](#) property to specify the row number before setting the RowMerge property.

Note You cannot merge rows if they have multiple levels. If the [ColumnLevels](#) property is set to a value greater than zero, the RowMerge property is automatically set to 0 (Off).

You might use this property to reconfigure database information. For example, assume your database contains schedule status for products that undergo a five-step process and the normal way to display that data is by product name (row 0) and process

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

status (rows 1-5).

Product	P1	P2	P3	P4	P5
Peppers					
Rice	C	C			
Corn	C	C	C	C	
Peas	C	C	C		
Beans	C	C	C	C	C

You could merge rows and display products that have completed all processes.

Product	P1	P2	P3	P4	P5
Peppers					
Rice	C				
Corn	C				
Peas	C				
Beans	C				

Use the [ColMerge](#) property to merge columns that contain the same text.

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a four-column list box control. Cells with the same text in the first and third rows are always merged. Cells with the same text in the second row are merged restrictively.

C

```
LC_SetColumns(hWnd, 4);
LC_SetLineStyle(hWnd, LC_LINESTYLE_LOWERED);
LC_SetColumnHeaderShow(hWnd, TRUE);
/* C - complete I - incomplete */
LC_SetInsertRow(hWnd, "Beans\tC\tC\tC");
LC_SetInsertRow(hWnd, "Pea\tC\tC\tI");
LC_SetInsertRow(hWnd, "Corn\tC\tI\tI");
LC_SetInsertRow(hWnd, "Rice\tI\tI\tI");
/* Define columns */
LC_SetCol(hWnd, 0);
LC_SetColHeaderText(hWnd, "Product");
LC_SetCol(hWnd, 1);
LC_SetColHeaderText(hWnd, "P1");
LC_SetCol(hWnd, 2);
LC_SetColHeaderText(hWnd, "P2");
LC_SetCol(hWnd, 3);
LC_SetColHeaderText(hWnd, "P3");
/* Merge the first three rows */
LC_SetRow(hWnd, 0);
LC_SetRowMerge(hWnd, LC_ROWMERGE_ALWAYS);
LC_SetRow(hWnd, 1);
LC_SetRowMerge(hWnd, LC_ROWMERGE_RESTRICTED);
LC_SetRow(hWnd, 2);
LC_SetRowMerge(hWnd, LC_ROWMERGE_ALWAYS);
```

C++

```
fpList1->SetColumns(4);
fpList1->SetLineStyle(LC_LINESTYLE_LOWERED);
fpList1->SetColumnHeaderShow(TRUE);
// C - complete I - incomplete
fpList1->SetInsertRow("Beans\tC\tC\tC");
fpList1->SetInsertRow("Peas\tC\tC\tI");
fpList1->SetInsertRow("Corn\tC\tI\tI");
fpList1->SetInsertRow("Rice\tI\tI\tI");
// Define columns
fpList1->SetCol(0);
fpList1->SetColHeaderText("Product");
fpList1->SetCol(1);
fpList1->SetColHeaderText("P1");
fpList1->SetCol(2);
fpList1->SetColHeaderText("P2");
fpList1->SetCol(3);
fpList1->SetColHeaderText("P3");
// Merge the first three rows
fpList1->SetRow(0);
fpList1->SetRowMerge(LC_ROWMERGE_ALWAYS);
fpList1->SetRow(1);
fpList1->SetRowMerge(LC_ROWMERGE_RESTRICTED);
fpList1->SetRow(2);
fpList1->SetRowMerge(LC_ROWMERGE_ALWAYS);
```

Visual Basic

```
fpList1.Columns = 4
fpList1.LineStyle = LC_LINESTYLE_LOWERED
fpList1.ColumnHeaderShow = True
' C - complete I - incomplete
fpList1.InsertRow = "Beans" & Chr$(9) & "C" & Chr$(9) & "C" & Chr$(9) & "C"
fpList1.InsertRow = "Peas" & Chr$(9) & "C" & Chr$(9) & "C" & Chr$(9) & "I"
```

```
fpList1.InsertRow = "Corn" & Chr$(9) & "C" & Chr$(9) & "I" & Chr$(9) & "I"
fpList1.InsertRow = "Rice" & Chr$(9) & "I" & Chr$(9) & "I" & Chr$(9) & "I"
' Define columns
fpList1.Col = 0
fpList1.ColHeaderText = "Product"
fpList1.Col = 1
fpList1.ColHeaderText = "P1"
fpList1.Col = 2
fpList1.ColHeaderText = "P2"
fpList1.Col = 3
fpList1.ColHeaderText = "P3"
' Merge the first three rows
fpList1.Row = 0
fpList1.RowMerge = LC_ROWMERGE_ALWAYS
fpList1.Row = 1
fpList1.RowMerge = LC_ROWMERGE_RESTRICTED
fpList1.Row = 2
fpList1.RowMerge = LC_ROWMERGE_ALWAYS
```

See Also

[Merging Columns and Rows](#)

[ColMerge](#), [ColumnLevels](#), [MergeAdjustView](#), [Row](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ScrollBarH Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether the horizontal scroll bar in an fpCombo or fpList control is displayed continuously or only when necessary.

Syntax

C **UINT** LC_GetScrollBarH(**HWND** hWnd, **short FAR** *pValue);
UINT LC_SetScrollBarH(**HWND** hWnd, **short** value);

C++ **short Class::**GetScrollBarH(**void**);
Class::SetScrollBarH(**short** value);

Visual Basic [form.]control.ScrollBarH[= setting%]

Designer Page

[Scroll subtab of the Appearance designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Show	Displays the horizontal scroll bar	LC_SCROLLBARH_SHOW
1 - Show When Needed	(Default) Displays the horizontal scroll bar when necessary and hides it when it is not necessary	LC_SCROLLBARH_SHOW_WHEN_NEEDED
2 - Show Disabled	Displays the horizontal scroll bar as disabled when it is not necessary	LC_SCROLLBARH_SHOW_DISABLED
3 - Hide	Does not display the horizontal scroll bar	LC_SCROLLBARH_HIDE

Data Type

Integer (Enumerated)

See Also

[Displaying and Customizing Scroll Bars](#)

[ScrollHinc](#), [ScrollHScale](#), [ScrollBarV](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ScrollBarV Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether the vertical scroll bar in an fpCombo or fpList control is displayed continuously or only when necessary.

Syntax

C **UINT** LC_GetScrollBarV(**HWND** hWnd, **short FAR** *pValue);
UINT LC_SetScrollBarV(**HWND** hWnd, **short** value);

C++ **short** **Class**::GetScrollBarV(**void**);
Class::SetScrollBarV(**short** value);

Visual Basic [form.]control.ScrollBarV[= setting%]

Designer Page

[Scroll subtab of the Appearance designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Show	Displays the vertical scroll bar	LC_SCROLLBARV_SHOW
1 - Show When Needed	(Default) Displays the vertical scroll bar when necessary and hides it when it is not necessary	LC_SCROLLBARV_SHOW_WHEN_NEEDED
2 - Show Disabled	Displays the vertical scroll bar as disabled when it is not necessary	LC_SCROLLBARV_SHOW_DISABLED
3 - Hide	Does not display the vertical scroll bar	LC_SCROLLBARV_HIDE

Data Type

Integer (Enumerated)

See Also

[Displaying and Customizing Scroll Bars](#)

[ScrollBarH](#) property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ScrollHInc Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the number of units for the amount the list box scrolls when the user moves the horizontal scroll box in an fpCombo or fpList control.

Syntax

C *UINT* LC_GetScrollHInc(*HWND* hWnd, *short FAR* *lpValue);
UINT LC_SetScrollHInc(*HWND* hWnd, *short* value);

C++ *short Class::*GetScrollHInc(*void*);
*Class::*SetScrollHInc(*short* value);

Visual Basic [form.]control.ScrollHInc = value%

Designer Page

[Scroll subtab of the Appearance designer page](#)

Remarks

This property works with the [ScrollHScale](#) property to determine the amount the list box scrolls when the user clicks the left and right scroll arrows. The ScrollHScale property determines the measurement unit used for the amount of movement.

The default value for the ScrollHInc property is 0. When the ScrollHInc property is set to 0, the following default values are assigned, depending on the value of the ScrollHScale property:

ScrollHScale property setting	Default scrolling increment
0 - Twips	200 twips
1 - Pixels	15 pixels
2 - Avg Char Width	1 character
3 - Max Char Width	1 character
4 - Column	Not applicable (Scrolls one column at a time.)

Data Type

Integer

See Also

[Displaying and Customizing Scroll Bars](#)

[ScrollBarH](#), [ScrollHScale](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ScrollHScale Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the unit of measurement for the amount the list box scrolls when the user moves the horizontal scroll box in an fpCombo or fpList control.

Syntax

C **UINT** LC_GetScrollHScale(**HWND** hWnd, **short FAR** *pValue);
UINT LC_SetScrollHScale(**HWND** hWnd, **short** value);

C++ **short Class::**GetScrollHScale(**void**);
Class::SetScrollHScale(**short** value);

Visual Basic [form.]control.ScrollHScale[= setting%]

Designer Page

[Scroll subtab of the Appearance designer page](#)

Remarks

The ScrollHScale property works in conjunction with the [ScrollHInc](#) property to determine the amount the list box scrolls. The ScrollHInc property determines the number of units scrolled.

Setting	Description	Constant
0 - Twips	Scrolls a specified number of twips as determined by the ScrollHInc property setting	LC_SCROLLHSCALE_TWIPS
1 - Pixels	Scrolls a specified number of pixels as determined by the ScrollHInc property setting	LC_SCROLLHSCALE_PIXELS
2 - Avg Char Width	(Default) Scrolls a specified number of characters based on the average character width of the current default font multiplied by the ScrollHInc property setting	LC_SCROLLHSCALE_AVG_CHAR
3 - Max Char Width	Scrolls a specified number of characters based on the maximum character width of the current default font multiplied by the ScrollHInc property setting	LC_SCROLLHSCALE_MAX_CHAR
4 - Column	Scrolls one column at a time if the Columns property is set to a value greater than 0 (If this value is set and the Columns property is set to 0, the average character width is used.)	LC_SCROLLHSCALE_COLUMN
5 - Top Group	Scrolls one group at a time based on the group arrangement of the top group if more than one level of groups exists	LC_SCROLLHSCALE_TOP_GROUP
6 - Bottom Group	Scrolls one group at a time based on the group	LC_SCROLLHSCALE_BOTTOM_GROUP

arrangement of the bottom
group if more than one
level of groups exists

Data Type

Integer (Enumerated)

See Also

[Displaying and Customizing Scroll Bars](#)

[Columns](#), [ScrollHinc](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

SearchIgnoreCase Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether case is ignored when searching an fpCombo or fpList control.

Syntax

C `UINT LC_GetSearchIgnoreCase(HWND hWnd, BOOL FAR *pValue);`
UINT LC_SetSearchIgnoreCase(**HWND** hWnd, **BOOL** value);

C++ `BOOL Class::GetSearchIgnoreCase(void);`
Class::SetSearchIgnoreCase(**BOOL** value);

Visual Basic `[form.]control.SearchIgnoreCase[= boolean%]`

Designer Page

[Search designer page](#)

Remarks

You can perform a search at run time by setting the [SearchText](#) property to the search string and setting the [Action](#) property to 0 (Search). Set the SearchIgnoreCase property to specify whether the control ignores case when searching.

The default value for the SearchIgnoreCase property is True, which means the control searches for characters regardless of case. For example, if the SearchIgnoreCase property is set to True and the user searches for "milwaukee", the search will find both "milwaukee" and "Milwaukee".

Data Type

Integer (Boolean)

See Also

[Searching for List Items](#)

[Action](#), [AutoSearch](#), [SearchIndex](#), [SearchMethod](#), [SearchText](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

SearchIndex Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the index of the currently selected list item. The property is available at run time only.

Syntax

C *UINT* LC_GetSearchIndex(*HWND* hWnd, *long FAR *lpValue*);
UINT LC_SetSearchIndex(*HWND* hWnd, *long value*);

C++ *longClass*::GetSearchIndex(*void*);
Class::SetSearchIndex(*long value*);

Visual Basic [form.]control.SearchIndex[= value&]

Designer Page

[Search designer page](#)

Remarks

You can perform a search at run time by setting the [SearchText](#) property to the search string and setting the [Action](#) property to 0 (Search). When the fpCombo or fpList control finds a match, the SearchIndex property contains the index number of the selected item.

The value of the SearchIndex property is 0 for the first list item. If the search is unsuccessful, the value of the SearchIndex

property is

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

 1.

Data Type

Integer (Long)

See Also

[Searching for List Items](#)

[Action](#), [AutoSearch](#), [SearchIgnoreCase](#), [SearchMethod](#), [SearchText](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

SearchMethod Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether an fpCombo or fpList control requires an exact match when searching for a list item.

Syntax

C **UINT** LC_GetSearchMethod(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetSearchMethod(**HWND** hWnd, **short** value);

C++ **short Class::**GetSearchMethod(**void**);
Class::SetSearchMethod(**short** value);

Visual Basic [form.]control.SearchMethod[= setting%]

Designer Page

[Search designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Exact Match	(Default) Searches for a value containing the characters entered and does nothing when an exact match cannot be found	LC_SEARCHMETHOD_EXACT_MATCH
1 - Greater or Equal	Searches for the next letter when an exact match cannot be found	LC_SEARCHMETHOD_GREATER_OR_EQUAL
2 - Partial Match	Searches for the characters specified by the SearchText property, but does not require an exact match	LC_SEARCHMETHOD_PARTIAL_MATCH

You can perform a search at run time by setting the [SearchText](#) property to the search string and setting the [Action](#) property to 0 (Search).

The [SearchIndex](#) property returns the index of the selected list item.

You can use setting 2 (Partial Match) to enable searching similar to that provided by the [AutoSearch](#) property. For example, you can design a text box that displays characters as the user types them and searches the list for those characters.

Data Type

Integer (Enumerated)

See Also

[Searching for List Items](#)

[Action](#), [AutoSearch](#), [SearchIgnoreCase](#), [SearchIndex](#), [SearchText](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

SearchText Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the characters searched for in an fpCombo or fpList control. This property is available at run time only.

Syntax

C **UINT** LC_GetSearchText(*HWND* hWnd, *LPSTR* buffer, *UINT* nBufferSize);
UINT LC_SetSearchText(*HWND* hWnd, *LPCSTR* value);

C++ **LPSTR** *Class*::GetSearchText(*LPSTR* buffer, *UINT* bufferSize);
Class::SetSearchText(*LPCSTR* value);

Visual Basic [form.]control.SearchText[= text\$]

Designer Page

[Search designer page](#)

Remarks

You can perform a search at run time by setting the SearchText property to the search string and setting the [Action](#) property to 0 (Search).

When the fpCombo or fpList control finds a match, the SearchIndex property contains the index number of the selected item.

Data Type

String

See Also

[Searching for List Items](#)

[Action](#), [AutoSearch](#), [SearchIgnoreCase](#), [SearchIndex](#), [SearchMethod](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

SelCount Property

[See Also](#)

Applies To

fpList control

Description

Returns the number of items selected in an fpList control that allows multiple selections. This property is available at run time only.

Syntax

C **UINT** LbxGetSelCount(*HWND* hWnd, *long FAR* *pValue);

C++ **long** CfpListBox::GetSelCount(**void**);

Visual Basic [*form.*]fpList1.SelCount

Remarks

The default value for the SelCount property is 0.

Use the [SelMax](#) property to limit the number of rows that can be selected.

To create an fpList control that allows multiple selections, set the [MultiSelect](#) property.

Data Type

Integer (Long)

See Also

[Accessing List Items](#)

[MultiSelect](#), [NextSel](#), [SelMax](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

SelDrawFocusRect Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether to draw a focus rectangle around a selection in an fpCombo or fpList control.

Syntax

C `UINT LC_GetSelDrawFocusRect(HWND hWnd, BOOL FAR *pValue);`
UINT LC_SetSelDrawFocusRect(**HWND** hWnd, **BOOL** value);

C++ `BOOL Class::GetSelDrawFocusRect(void);`
Class::SetSelDrawFocusRect(**BOOL** value);

Visual Basic `[form.]control.SelDrawFocusRect[= boolean%]`

Designer Page

[Miscellaneous designer page](#)

Remarks

The default value for the SelDrawFocusRect property is True.

Setting the SelDrawFocusRect property to False hides the focus rectangle when a list item is selected, which may look better with three-dimensional effects (setting the [LineStyle](#) property to 3 (Lowered), 4 (Raised), 5 (Lowered w/ Line), or 6 (Raised w/ Line) and the [ListApplyTo](#) property to 4 (Sel Rows)).

Data Type

Integer (Boolean)

See Also

[Hiding the Focus Rectangle Around Selected Items](#)

[LineStyle](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Selected Property

[See Also](#)

Applies To

fpList control

Description

Sets or returns the selection status of an item in an fpList control. This property is available at run time only.

Syntax

C **UINT** LbxGetSelected(**HWND** hWnd, **BOOL FAR** *lpValue);
 UINT LbxSetSelected(**HWND** hWnd, **BOOL** value);

C++ **BOOL**CfpListBox::GetSelected(**void**);
 CfpListBox::SetSelected(**BOOL** value);

Visual Basic [form].fpList1.Selected[= boolean%]

Remarks

The default value for the Selected property is False.

When the Selected property is set to True, the item specified by the [Row](#) property is selected (highlighted).

Before setting the Selected property, you must specify a row with the Row property.

Note that the List Pro Selected property is not an array property as it is in Visual Basic. For more information, refer to the Selected property in the Visual Basic documentation.

Data Type

Integer (Boolean)

See Also

[Accessing List Items](#)

[List](#), [ListIndex](#), [MultiSelect](#), [NextSel](#), [Row](#), [TopIndex](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

SelLength Property

[See Also](#)

[Example](#)

Applies To

fpCombo control

Description

Sets or returns the number of characters selected in the edit field. This property is available at run time only.

Syntax

C *UINT* CbxGetSelLength(*HWND* hWnd, *long FAR *lpValue*);
 UINT CbxSetSelLength(*HWND* hWnd, *long* value);

C++ *long* CfpComboBox::GetSelLength(*void*);
 CfpComboBox::SetSelLength(*long* value);

Visual Basic [form.]fpCombo1.SelLength[= value&]

Remarks

Setting the SelLength property to a negative value causes a run-time error.

You must move the focus to the fpCombo control (using the Forms Activate event or the SetFocus method) before setting the SelLength property.

Refer to the Visual Basic documentation for additional information about this property.

Data Type

Integer (Long)

[Print](#)

[Copy](#)

[Close](#)

The following example allows the user to search for a specified word using an fpCombo control, a text control, and a button control. This example searches for a specified word and, if the word is found, positions it to be visible in the control.

Visual Basic

```
Sub Form_Load ( )
fpCombo1.Text = "Search this text to see if there is some matching text within this
control."
End Sub

Sub Command_Click ( )
Dim x As Integer
Dim Length As Integer
' Check for selected text and retrieve length of found text
x = InStr(fpCombo1.Text, Text1.Text)
Length = Len(fpCombo1.Text)
' If no matching text is found, display message and exit
If x = 0 Then
    MsgBox "No matching text was found!"
    Exit Sub
End If
' Select matching text
fpCombo1.SetFocus
fpCombo1.SelStart = x - 1
fpCombo1.SelLength = Len(Text1.Text)
End Sub
```

See Also

[SelStart](#), [SelText](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

SelMax Property

[See Also](#) [Example](#)

Applies To

fpList control

Description

Sets or returns the maximum number of rows that can be selected in an fpList control.

Syntax

C *UINT* LbxGetSelMax(*HWND* hWnd, *long FAR* *lpValue);
 UINT LbxSetSelMax(*HWND* hWnd, *long* value);

C++ *long*CfpListBox::GetSelMax(*void*);
 CfpListBox::SetSelMax(*long* value);

Visual Basic [form].fpList1.SelMax[= value&]

Designer Page

[Miscellaneous designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the SelMax property is 1, which lets the user make any number of selections. Use the [SelCount](#) property to return the actual number of rows selected.

Data Type

Integer (Long)

See Also

[Accessing List Items](#)

[NextSel](#), [SelCount](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

SelStart Property

[See Also](#) [Example](#)

Applies To

fpCombo control

Description

Sets or returns the starting point of selected characters or the position of the insertion point if no characters are selected in the edit field. This property is available at run time only.

Syntax

C	UINT CbxGetSelStart(HWND hWnd, long FAR *pValue); UINT CbxSetSelStart(HWND hWnd, long value);
C++	long CfpComboBox::GetSelStart(void); CfpComboBox::SetSelStart(long value);
Visual Basic	[form.]fpCombo1.SelStart[= value&]

Remarks

The value of the SelStart property is based on the number of characters in the edit field. The first position is zero.

You must move the focus to the fpCombo control (using the Forms Activate event or the SetFocus method) before setting the SelStart property.

Visual Basic users can refer to the Visual Basic documentation for additional information about this property.

Data Type

Integer (Long)

See Also

[SelLength](#), [SelText](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

SelText Property

[See Also](#)

Applies To

fpCombo control

Description

Sets or returns the string containing the currently selected text in the edit field. This property is available at run time only.

Syntax

C **UINT** CbxGetSelText(**HWND** hWnd, **LPSTR** buffer, **UINT** nBufferSize);
UINT CbxSetSelText(**HWND** hWnd, **LPCSTR** value);

C++ **LPSTR** CfpComboBox::GetSelText(**LPSTR** buffer, **UINT** bufferSize);
CfpComboBox::SetSelText(**LPCSTR** value);

Visual Basic [form.]fpCombo1.SelText[= text\$]

Remarks

When no selection has been made in the control, the SelText property returns an empty string.

You can assign a string to the SelText property to replace the selected text.

You must move the focus to the fpCombo control (using the Forms Activate event or the SetFocus method) before setting the SelText property.

Visual Basic users can refer to the Visual Basic documentation for additional information about this property.

Data Type

String

See Also

[SelLength](#), [SelStart](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Sorted Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the type of sort performed on the contents of a single-column fpCombo or fpList control.

Syntax

C **UINT** LC_GetSorted(**HWND** hWnd, **short FAR** *pValue);
UINT LC_SetSorted(**HWND** hWnd, **short** value);

C++ **short Class::**GetSorted(**void**);
Class::SetSorted(**short** value);

Visual Basic [form.]control.Sorted[= setting%]

Designer Page

[Sort designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - None	(Default) Removes sorting from the column	LC_SORTED_NONE
1 - Ascending	Sorts from the beginning of the alphabet	LC_SORTED_ASCENDING
2 - Descending	Sorts from the end of the alphabet	LC_SORTED_DESCENDING

If you use the Sorted property on a multiple-column control, only the first column is sorted. Use the [ColSorted](#) property to perform a sort on multiple-column List Pro controls.

Data Type

Integer (Enumerated)

See Also

[Sorting List Items](#)

[ColSorted](#), [ColSortSeq](#), [SortState](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

SortState Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether sorting is temporarily turned off when adding items to a sorted list in an fpCombo or fpList control. This property is available at run time only.

Syntax

C **UINT** LC_GetSortState(**HWND** hWnd, **short FAR** *pValue);
UINT LC_SetSortState(**HWND** hWnd, **short** value);

C++ **short Class::**GetSortState(**void**);
Class::SetSortState(**short** value);

Visual Basic [form.]control.SortState[= setting%]

Designer Page

[Sort designer page](#)

Remarks

The SortState property greatly improves performance when adding large numbers of items or rows to a sorted list. When you add an item to a sorted list, the fpCombo or fpList control inserts the item in the correct order. This process can take a long time when the list is long and you are adding many items. To improve performance, set the SortState property to 2 (Suspend) before adding the items. After adding the items, reset the SortState property to 1 (Active (Re-sort)) to sort the list immediately. If the list does not need to be re-sorted until the next item is added, set the SortState property to 0 (Active).

The following settings are available:

Setting	Description	Constant
0 - Active	(Default) Turns sorting on without immediately re-sorting the list (the list is sorted when the next item is added)	LC_SORTSTATE_ACTIVE
1 - Active (Re-sort)	Re-sorts the list quickly	LC_SORTSTATE_ACTIVE_RESORT
2 - Suspend	Turns off sorting temporarily until SortState property is reset to 0 or 1	LC_SORTSTATE_ACTIVE_SUSPEND

Data Type

Integer (Enumerated)

See Also

[Sorting List Items](#)

[Col](#), [ColSorted](#), [ColSortSeq](#), [Sorted](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Style Property

[See Also](#) [Example](#)

Applies To

fpCombo control

Description

Sets or returns the type of combo box displayed by an fpCombo control.

Syntax

C **UINT** CbxGetStyle(*HWND* hWnd, **short FAR** *pValue);
UINT CbxSetStyle(*HWND* hWnd, **short** value);

C++ **short** CfpComboBox::GetStyle(**void**);
CfpComboBox::SetStyle(**short** value);

Visual Basic [form.]fpCombo1.Style[= *setting*%]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Drop-Down Combo	(Default) Creates a drop-down list and an edit field	CBX_STYLE_DROPDOWN_COMBO
1 - Simple Combo	Creates an edit field and a standard list (like the fpList control)	CBX_STYLE_SIMPLE_COMBO
2 - Drop-Down List	Creates a drop-down list with a static field	CBX_STYLE_DROPDOWN_LIST

Note If you choose 1 (Simple Combo), you must resize the control so that it is large enough to display the list portion.

For more information on each style, refer to the Style property in the Visual Basic documentation.

Data Type

Integer (Enumerated)

See Also

[Choosing the fpCombo Control Style](#)

[DropDown](#) event

[Text](#) property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Text Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the contents of the selected row in a single-column control. For the fpList control the Text property is available at run time only.

Syntax

C	UINT LC_GetText(HWND hWnd, LPSTR buffer, UINT nBufferSize); UINT CbxSetText(HWND hWnd, LPCSTR value);
C++	LPSTR Class ::GetText(LPSTR buffer, UINT bufferSize); Class ::SetText(LPCSTR value);
Visual Basic	[form.]control.Text[= text\$]

Designer Page

[Add Data designer page](#) (for fpCombo control only)

Remarks

For the fpCombo control, when the [Style](#) property is set to 0 (Drop-Down Combo) or 1 (Simple Combo), the Text property returns the text contained in the edit (or static) field. When the Style property is set to 2 (Drop-Down List), the Text property returns the value of the selected list item.

For the fpList control, the Text property sets or returns the text for the currently selected row.

Visual Basic users can refer to the Visual Basic documentation for additional information about this property.

Data Type

String

See Also

[Accessing List Items](#)

[SelLength](#), [SelStart](#), [SelText](#), [Style](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

TextOrientation Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether the text and graphic displayed in the control are rotated 90, 180, or 270 degrees or whether the text is displayed vertically.

Syntax

C **UINT** LC_GetTextOrientation(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetTextOrientation(**HWND** hWnd, **short** value);

C++ **short Class::**GetTextOrientation(**void**);
Class::SetTextOrientation(**short** value);

Visual Basic [form.]control.TextOrientation[= setting%]

Designer Page

[List subtab of the ApplyTo designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Default	Uses the text orientation setting of the hierarchical predecessor	LC_TEXTORIENTATION_DEFAULT
1 - Horizontal	(Default) Displays text and graphics horizontally within each cell	LC_TEXTORIENTATION_HORIZONTAL
2 - Vertical LTR	Displays text vertically within each cell and wraps from left to right	LC_TEXTORIENTATION_VERTICAL_LTR
3 - Rotate Down	Rotates text and graphics 90 degrees (to "3 o'clock")	LC_TEXTORIENTATION_ROTATE_DOWN
4 - Rotate Up	Rotates text and graphics 270 degrees (to "9 o'clock")	LC_TEXTORIENTATION_ROTATE_UP
5 - Invert	Rotates text and graphics 180 degrees (to "6 o'clock")	LC_TEXTORIENTATION_INVERT
6 - Vertical RTL	Displays text vertically within each cell and wraps from right to left	LC_TEXTORIENTATION_VERTICAL_RTL

If the selected font does not rotate, the fpCombo or fpList control selects a similar, TrueType font that does rotate.

If you set the TextOrientation property to 1 (Vertical LTR) or 6 (Vertical RTL) and the [MultiLine](#) property is set to True, the text in the list is displayed vertically from top to bottom and wraps from left to right (LTR) or right to left (RTL).

You might want to use setting 6 (Vertical RTL) for displaying text in languages that are written vertically and read from right to left, such as Japanese. If you choose setting 6 (Vertical RTL), note that if the header text wraps, it wraps to the left (text runs top to bottom, right to left).

You can use the [ListApplyTo](#) property to specify where the TextOrientation property applies.

Tip If you set the TextOrientation property to 1 (Vertical LTR) or 6 (Vertical RTL) and you are displaying graphics in the list item, set the [PictureAlignH](#) property to 4 (Left of Text) or 5 (Right of Text) and the [PictureAlignV](#) property to 4 (Top of Text) or 5 (Bottom of Text) for best results.

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a two-column list box with the first column merged. The text in the first column is displayed vertically from right to left.

C

```
LC_SetColumns(hWnd, 2);
LC_SetColumnHeaderShow(hWnd, TRUE);
LC_SetLineStyle(hWnd, LC_LISTSTYLE_LOWERED);
LC_SetInsertRow(hWnd, "NY" & Chr$(9) & "Yankees");
LC_SetInsertRow(hWnd, "NY" & Chr$(9) & "Mets");
LC_SetInsertRow(hWnd, "CA" & Chr$(9) & "Padres");
LC_SetInsertRow(hWnd, "CA" & Chr$(9) & "Angels");
LC_SetInsertRow(hWnd, "CA" & Chr$(9) & "Giants");
LC_SetInsertRow(hWnd, "CA" & Chr$(9) & "Dodgers");
/* Define columns */
LC_SetCol(hWnd, 0);
LC_SetColID(hWnd, 11);
LC_SetCol(hWnd, 1);
LC_SetColID(hWnd, 22);
/* Merge first column and define column text */
LC_SetColFromID(hWnd, 11);
LC_SetColMerge(hWnd, 1);
LC_SetColHeaderText(hWnd, "State");
LC_SetColFromID(hWnd, 22);
LC_SetColHeaderText(hWnd, "Team");
/* Orient text in column headers horizontally */
LC_SetListApplyTo(hWnd, 7);
LC_SetTextOrientation(hWnd, LC_TEXTORIENTATION_HORIZONTAL);
/* Orient text in first column vertically RTL */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SINGLE_ITEM);
LC_SetColFromID(hWnd, 11);
LC_SetTextOrientation(hWnd, LC_TEXTORIENTATION_VERTICAL_RTL);
```

C++

```
fpList1->SetColumns(2);
fpList1->SetColumnHeaderShow(TRUE);
fpList1->SetLineStyle(LC_LISTSTYLE_LOWERED);
fpList1->SetInsertRow("NY" & Chr$(9) & "Yankees");
fpList1->SetInsertRow("NY" & Chr$(9) & "Mets");
fpList1->SetInsertRow("CA" & Chr$(9) & "Padres");
fpList1->SetInsertRow("CA" & Chr$(9) & "Angels");
fpList1->SetInsertRow("CA" & Chr$(9) & "Giants");
fpList1->SetInsertRow("CA" & Chr$(9) & "Dodgers");
// Define columns
fpList1->SetCol(0);
fpList1->SetColID(11);
fpList1->SetCol(1);
fpList1->SetColID(22);
// Merge first column and define column text
fpList1->SetColFromID(11);
fpList1->SetColMerge(1);
fpList1->SetColHeaderText("State");
fpList1->SetColFromID(22);
fpList1->SetColHeaderText("Team");
// Orient text in column headers horizontally
fpList1->SetListApplyTo(7);
fpList1->SetTextOrientation(LC_TEXTORIENTATION_HORIZONTAL);
// Orient text in first column vertically RTL
fpList1->SetListApplyTo(LC_LISTAPPLYTO_SINGLE_ITEM);
fpList1->SetColFromID(11);
fpList1->SetTextOrientation(LC_TEXTORIENTATION_VERTICAL_RTL);
```

Visual Basic

```
fpList1.Columns = 2
fpList1.ColumnHeaderShow = True
fpList1.LineStyle = 3 'LC_LINESTYLE_LOWERED
fpList1.InsertRow = "NY" & Chr$(9) & "Yankees"
fpList1.InsertRow = "NY" & Chr$(9) & "Mets"
fpList1.InsertRow = "CA" & Chr$(9) & "Padres"
fpList1.InsertRow = "CA" & Chr$(9) & "Angels"
fpList1.InsertRow = "CA" & Chr$(9) & "Giants"
fpList1.InsertRow = "CA" & Chr$(9) & "Dodgers"
' Define columns
fpList1.Col = 0
fpList1.ColID = 11
fpList1.Col = 1
fpList1.ColID = 22
' Merge first column and define column text
fpList1.ColFromID = 11
fpList1.ColMerge = 1
fpList1.ColHeaderText = "State"
fpList1.ColFromID = 22
fpList1.ColHeaderText = "Team"
' Orient text in column headers horizontally
fpList1.ListApplyTo = 7
fpList1.TextOrientation = LC_TEXTORIENTATION_HORIZONTAL
' Orient text in first column vertically RTL
fpList1.ListApplyTo = LC_LISTAPPLYTO_SINGLE_ITEM
fpList1.ColFromID = 11
fpList1.TextOrientation = LC_TEXTORIENTATION_VERTICAL_RTL
```


See Also

[Orienting Text and Graphics](#)

[ListApplyTo](#), [MultiLine](#), [PictureAlignH](#), [PictureAlignV](#), [Text](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ThreeDFrameColor Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Set or return the frame color.

Syntax

C **UINT** LC_GetThreeDFrameColor(**HWND** hWnd, **COLORREF FAR** *lpValue);
UINT LC_SetThreeDFrameColor(**HWND** hWnd, **COLORREF** value);

C++ **COLORREF** *Class*::GetThreeDFrameColor(**void**);
Class::SetThreeDFrameColor(**COLORREF** value);

Visual Basic [form.]control.ThreeDFrameColor[= color]

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The default value for the ThreeDFrameColor property is &H8000000F& (Windows system button face color).

To add three-dimensional effects to a control, you can create inner and outer three-dimensional borders by setting the [ThreeDInsideStyle](#) and [ThreeDOutsideStyle](#) properties. The [ThreeDInsideWidth](#) and [ThreeDOutsideWidth](#) properties let you adjust the width of the three-dimensional borders.

You can create a frame to add space between the inner and outer three-dimensional borders. To create a frame, you must set both the ThreeDInsideStyle and ThreeDOutsideStyle properties to values other than 0 (None). Then set the ThreeDFrameWidth property to the width of the frame. For a small (default-sized) control, a frame width of 2 or 3 pixels looks best.

Data Type

Color

See Also

[Creating a Frame](#)

[Appearance](#), [ThreeDFrameWidth](#), [ThreeDInsideHighlightColor](#), [ThreeDInsideShadowColor](#), [ThreeDInsideStyle](#), [ThreeDInsideWidth](#), [ThreeDOutsideHighlightColor](#), [ThreeDOutsideShadowColor](#), [ThreeDOutsideStyle](#), [ThreeDOutsideWidth](#)
properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ThreeDFrameWidth, ThreeDInsideWidth, ThreeDOutsideWidth Properties

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Set or return the inner and outer border widths and the distance between the inner and outer borders, in pixels.

Syntax

C *UINT* LC_GetThreeDFrameWidth(*HWND* hWnd, *short FAR* *lpValue);
UINT LC_SetThreeDFrameWidth(*HWND* hWnd, *short* value);

C++ *short Class*::GetThreeDFrameWidth(*void*);
Class::SetThreeDFrameWidth(*short* value);

Visual Basic [form.]control.ThreeDFrameWidth[= value%]

Note The ThreeDFrameWidth, ThreeDInsideWidth, and ThreeDOutsideWidth properties use the same syntax.

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The default value for the ThreeDFrameWidth property is 0. The default value for the ThreeDInsideWidth and ThreeDOutsideWidth properties is 1 pixel.

To add three-dimensional effects to a control, you can create inner and outer borders by setting the [ThreeDInsideStyle](#) and [ThreeDOutsideStyle](#) properties. The ThreeDInsideWidth and ThreeDOutsideWidth properties let you adjust the width of the three-dimensional borders. Customize the colors displayed in the inner and outer borders by setting the [ThreeDInsideHighlightColor](#), [ThreeDInsideShadowColor](#), [ThreeDOutsideHighlightColor](#), and [ThreeDOutsideShadowColor](#) properties.

You can create a frame to add space between the inner and outer three-dimensional borders. To create a frame, you must set both the ThreeDInsideStyle and ThreeDOutsideStyle properties to values other than 0 (None). Then set the ThreeDFrameWidth property to the width of the frame. For a small (default-sized) control, a frame width of 2 or 3 pixels looks best.

For more information about customizing your control's borders, see [Customizing the Control's Borders](#). For more information about creating a frame for your control, see [Creating a Frame](#).

You can specify a predefined border appearance for your control that automatically sets these and other properties by using the [Appearance](#) property. For instructions and a description of the predefined appearance styles, see [Using Predefined Appearance Styles](#).

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example demonstrates how to create a frame for an fpList control.

C

```
LC_SetThreeDFrameWidth(hWnd, 3);
/* red RGB(255, 0, 0) */
LC_SetThreeDFrameColor(hWnd, 0x000000FF);
/* Define inside 3D look */
LC_SetThreeDInsideStyle(hWnd, LC_THREEDINSIDESTYLE_RAISED);
LC_SetThreeDInsideWidth(hWnd, 4);
/* yellow RGB(255, 255, 0) */
LC_SetThreeDInsideHighlightColor(hWnd, 0x0000FFFF);
/* blue RGB(0, 0, 255) */
LC_SetThreeDInsideShadowColor(hWnd, 0x00FF0000);
/* Define outside 3D look */
LC_SetThreeDOutsideStyle(hWnd, LC_THREEDOUTSIDESTYLE_LOWERED);
LC_SetThreeDOutsideWidth(hWnd, 4);
/* blue RGB(0, 0, 255) */
LC_SetThreeDOutsideHighlightColor(hWnd, 0x00FF0000);
/* yellow RGB(0, 255, 255) */
LC_SetThreeDOutsideShadowColor(hWnd, 0x00FFFF00);
```

C++

```
fpList1->SetThreeDFrameWidth(3);
// red RGB(255, 0, 0)
fpList1->SetThreeDFrameColor(0x000000FF);
// Define inside 3D look
fpList1->SetThreeDInsideStyle(LC_THREEDINSIDESTYLE_RAISED);
fpList1->SetThreeDInsideWidth(4);
// yellow RGB(0, 255, 255)
fpList1->SetThreeDInsideHighlightColor(0x00FFFF00);
// blue RGB(0, 0, 255)
fpList1->SetThreeDInsideShadowColor(0x00FF0000);
// Define outside 3D look
fpList1->SetThreeDOutsideStyle(LC_THREEDOUTSIDESTYLE_LOWERED);
fpList1->SetThreeDOutsideWidth(4);
// blue RGB(0, 0, 255)
fpList1->SetThreeDOutsideHighlightColor(0x00FF0000);
// yellow RGB(0, 255, 255)
fpList1->SetThreeDOutsideShadowColor(0x00FFFF00);
```

Visual Basic

```
fpList1.ThreeDFrameWidth = 3
' red RGB(255, 0, 0)
fpList1.ThreeDFrameColor = &H000000FF&
' Define inside 3D look
fpList1.ThreeDInsideStyle = LC_THREEDINSIDESTYLE_RAISED
fpList1.ThreeDInsideWidth = 4
' yellow RGB(0, 255, 255)
fpList1.ThreeDInsideHighlightColor = &H00FFFF00&
' blue RGB(0, 0, 255)
fpList1.ThreeDInsideShadowColor = &H00FF0000&
' Define outside 3D look
fpList1.ThreeDOutsideStyle = LC_THREEDOUTSIDESTYLE_LOWERED
fpList1.ThreeDOutsideWidth = 4
' blue RGB(0, 0, 255)
fpList1.ThreeDOutsideHighlightColor = &H00FF0000&
' yellow RGB(0, 255, 255)
fpList1.ThreeDOutsideShadowColor = &H00FFFF00&
```

See Also

[Creating a Frame](#)

[Specifying the Border Appearance](#)

[Using Predefined Appearance Styles](#)

[Appearance](#), [ThreeDFrameColor](#), [ThreeDInsideHighlightColor](#), [ThreeDInsideShadowColor](#), [ThreeDInsideStyle](#), [ThreeDOutsideHighlightColor](#), [ThreeDOutsideShadowColor](#), [ThreeDOutsideStyle](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ThreeDInsideHighlightColor, ThreeDOutsideHighlightColor Properties

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Set or return the three-dimensional highlight colors of the control's inner and outer borders.

Syntax

C **UINT** LC_GetThreeDInsideHighlightColor(**HWND** hWnd, **COLORREF FAR** *lpValue);
UINT LC_SetThreeDInsideHighlightColor(**HWND** hWnd, **COLORREF** value);

C++ **COLORREF Class**::GetThreeDInsideHighlightColor(**void**);
Class::SetThreeDInsideHighlightColor(**COLORREF** value);

Visual Basic [form.]control.ThreeDInsideHighlightColor[= color]

Note The ThreeDInsideHighlightColor and ThreeDOutsideHighlightColor properties use the same syntax.

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The default values for these properties are &H8000000F& (Windows system button face color) for ThreeDInsideHighlightColor and &H80000014& (Windows system button highlight color) for ThreeDOutsideHighlightColor.

For these properties to have an effect, the [ThreeDInsideStyle](#) and [ThreeDOutsideStyle](#) properties must be set to values other than 0 (None).

The ThreeDInsideHighlightColor is the color displayed for the highlight portion of the inner, three-dimensional border. The ThreeDOutsideHighlightColor is the color displayed for the highlight portion of the outer, three-dimensional border.

To add three-dimensional effects to a control, you can create inner and outer borders by setting the ThreeDInsideStyle and ThreeDOutsideStyle properties. The [ThreeDInsideWidth](#) and [ThreeDOutsideWidth](#) properties let you adjust the width of the three-dimensional borders. Customize the highlight colors displayed for the inner and outer borders by setting the ThreeDInsideHighlightColor and ThreeDOutsideHighlightColor properties.

For more information about customizing your control's borders, see [Customizing the Control's Borders](#).

You can specify a predefined border appearance for your control that automatically sets these and other properties by using the [Appearance](#) property. For instructions and a description of the predefined appearance styles, see [Using Predefined Appearance Styles](#).

Note Be aware of the control's and the form's background colors when choosing the settings for the ThreeDInsideHighlightColor and ThreeDOutsideHighlightColor properties. For example, if you set the ThreeDOutsideHighlightColor property to white, and the background color of the form is white, the highlight color will be indistinguishable from the background.

Data Type

Color

See Also

[Specifying the Border Appearance](#)

[Using Predefined Appearance Styles](#)

[Appearance](#), [ThreeDFrameColor](#), [ThreeDFrameWidth](#), [ThreeDInsideShadowColor](#), [ThreeDInsideStyle](#), [ThreeDInsideWidth](#), [ThreeDOutsideShadowColor](#), [ThreeDOutsideStyle](#), [ThreeDOutsideWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ThreeDInsideShadowColor, ThreeDOutsideShadowColor Properties

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Set or return the three-dimensional shadow colors of the control's inner and outer borders.

Syntax

C **UINT** LC_GetThreeDInsideShadowColor(**HWND** hWnd, **COLORREF FAR** *lpValue);
UINT LC_SetThreeDInsideShadowColor(**HWND** hWnd, **COLORREF** value);

C++ **COLORREF Class**::GetThreeDInsideShadowColor(**void**);
Class::SetThreeDInsideShadowColor(**COLORREF** value);

Visual Basic [form.]control.ThreeDInsideShadowColor[= color]

Note The ThreeDInsideShadowColor and ThreeDOutsideShadowColor properties use the same syntax.

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The default values for these properties are &H80000006& (Windows system window frame color) for ThreeDInsideShadowColor and &H80000010& (Windows system button shadow color) for ThreeDOutsideShadowColor.

For these properties to have an effect, the [ThreeDInsideStyle](#) and [ThreeDOutsideStyle](#) properties must be set to values other than 0 (None).

To add three-dimensional effects to a control, you can create inner and outer borders by setting the ThreeDInsideStyle and ThreeDOutsideStyle properties. The [ThreeDInsideWidth](#) and [ThreeDOutsideWidth](#) properties let you adjust the width of the three-dimensional borders. Customize the shadow colors displayed for the inner and outer borders by setting the ThreeDInsideShadowColor and ThreeDOutsideShadowColor properties.

For more information about customizing your control's borders, see [Customizing the Control's Borders](#).

You can specify a predefined border appearance for your control that automatically sets these and other properties by using the [Appearance](#) property. For instructions and a description of the predefined appearance styles, see [Using Predefined Appearance Styles](#).

Note Be aware of the control's and the form's background colors when choosing the settings for the ThreeDInsideShadowColor and ThreeDOutsideShadowColor properties. For example, if you set the ThreeDOutsideShadowColor property to white, and the background color of the form is white, the shadow color will be indistinguishable from the background.

Data Type

Color

See Also

[Specifying the Border Appearance](#)

[Using Predefined Appearance Styles](#)

[Appearance](#), [ThreeDFrameColor](#), [ThreeDFrameWidth](#), [ThreeDInsideHighlightColor](#), [ThreeDInsideStyle](#), [ThreeDInsideWidth](#), [ThreeDOutsideHighlightColor](#), [ThreeDOutsideStyle](#), [ThreeDOutsideWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ThreeDInsideStyle, ThreeDOutsideStyle Properties

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Set or return the three-dimensional style of the control's inner and outer borders.

Syntax

C **UINT** LC_GetThreeDInsideStyle(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetThreeDInsideStyle(**HWND** hWnd, **short** value);

C++ **short Class::**GetThreeDInsideStyle(**void**);
Class::SetThreeDInsideStyle(**short** value);

Visual Basic [form.]control.ThreeDInsideStyle[= setting%]

Note The ThreeDInsideStyle and ThreeDOutsideStyle properties use the same syntax.

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

To add three-dimensional effects to a control, you can create inner and outer borders by setting these properties. The [ThreeDInsideWidth](#) and [ThreeDOutsideWidth](#) properties let you adjust the width of the three-dimensional borders. Customize the colors displayed in the inner and outer borders by setting the [ThreeDInsideHighlightColor](#), [ThreeDInsideShadowColor](#), [ThreeDOutsideHighlightColor](#), and [ThreeDOutsideShadowColor](#) properties.

The following settings are available:

Setting	Description	Constants
0 - None	(Default) No three-dimensional appearance	LC_THREEDINSIDESTYLE_NONE LC_THREEDOUTSIDESTYLE_NONE
1 - Lowered	Shadow color used to color the top and left border sides, highlight color used to color the bottom and right border sides	LC_THREEDINSIDESTYLE_LOWERED LC_THREEDOUTSIDESTYLE_LOWERED
2 - Raised	Shadow color used to color the bottom and right border sides, highlight color used to color the top and left border sides	LC_THREEDINSIDESTYLE_RAISED LC_THREEDOUTSIDESTYLE_RAISED

You can create a frame by adding space between the inner and outer three-dimensional borders. To create a frame, you must set both the ThreeDInsideStyle and ThreeDOutsideStyle properties to values other than 0 (None). Then set the [ThreeDFrameWidth](#) property to specify the width of the frame. For a small (default-sized) control, a frame width of 2 or 3 pixels looks best.

For more information about customizing your control's borders, see [Customizing the Control's Borders](#). For more information about creating a frame for your control, see [Creating a Frame](#).

You can specify a predefined border appearance for your control that automatically sets these and other properties by using the [Appearance](#) property. For instructions and a description of the predefined appearance styles, see [Using Predefined Appearance Styles](#).

Data Type

Integer (Enumerated)

See Also

[Creating a Frame](#)

[Customizing the Control's Borders](#)

[Specifying the Border Appearance](#)

[Using Predefined Appearance Styles](#)

[Appearance](#), [ThreeDFrameColor](#), [ThreeDFrameWidth](#), [ThreeDInsideHighlightColor](#), [ThreeDInsideShadowColor](#), [ThreeDInsideWidth](#), [ThreeDOnFocusInvert](#), [ThreeDOutsideHighlightColor](#), [ThreeDOutsideShadowColor](#), [ThreeDOutsideWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ThreeDOnFocusInvert Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether to invert the three-dimensional border colors when a control receives the focus.

Syntax

C **UINT** LC_GetThreeDOnFocusInvert(**HWND** hWnd, **BOOL FAR** *lpValue);
 UINT LC_SetThreeDOnFocusInvert(**HWND** hWnd, **BOOL** value);

C++ **BOOL Class::**GetThreeDOnFocusInvert(**void**);
 Class::SetThreeDOnFocusInvert(**BOOL** value);

Visual Basic [form.]control.ThreeDOnFocusInvert[= boolean%]

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The default value for the ThreeDOnFocusInvert property is False.

When the ThreeDOnFocusInvert property is set to True, the three-dimensional effect is inverted while the control has the focus. That is, the highlight and shadow colors are switched.

For the ThreeDOnFocusInvert property to have an effect, the [ThreeDInsideStyle](#) or [ThreeDOutsideStyle](#) property must be set to a value other than 0 (None), and the Enabled property must be set to True.

Data Type

Integer (Boolean)

[Print](#)

[Copy](#)

[Close](#)

The following example creates an fpList control that inverts the three-dimensional effect when it receives the focus. The outer three-dimensional style is lowered and the outer border width is 2 pixels.

C

```
/* Invert 3D when in focus */
LC_SetThreeDOnFocusInvert(hWnd, TRUE);
/* Lowered 3D style */
LC_SetThreeDOutsideStyle(hWnd, LC_THREEDOUTSIDESTYLE_LOWERED);
/* Outside border 3D width */
LC_SetThreeDOutsideWidth(hWnd, 2);
```

C++

```
// Invert 3D when in focus
fpList1->SetThreeDOnFocusInvert(TRUE);
// Lowered 3D style
fpList1->SetThreeDOutsideStyle(LC_THREEDOUTSIDESTYLE_LOWERED);
// Outside border 3D width
fpList1->SetThreeDOutsideWidth(2);
```

Visual Basic

```
' Invert 3D when in focus
fpList1.ThreeDOnFocusInvert = True
' Lowered 3D style
fpList1.ThreeDOutsideStyle = LC_THREEDOUTSIDESTYLE_LOWERED
' Outside border 3D width
fpList1.ThreeDOutsideWidth = 2
```

See Also

[Displaying Focus on the Control](#)

[BorderDropShadow](#), [ThreeDInsideStyle](#), [ThreeDOutsideStyle](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

TopIndex Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the index of the item at the top of the list in an fpCombo or fpList control. This property is available at run time only.

Syntax

C	UINT LC_GetTopIndex(<i>HWND</i> hWnd, long FAR *lpValue); UINT LC_SetTopIndex(<i>HWND</i> hWnd, long value);
C++	longClass:: GetTopIndex(void); Class:: SetTopIndex(long value);
Visual Basic	[form.]control.TopIndex[= value&]

Remarks

All items in an fpList control or in the list portion of an fpCombo control are numbered according to their current positions. The first item has an index number of 0, the second item has an index of 1, and so on. The TopIndex property returns the index of the item currently displayed at the top of the list. If the user scrolls down the list, the TopIndex property tells you which item is shown at the top, regardless of which item is selected.

The default value for the TopIndex property is 0.

Data Type

Integer (Long)

[Print](#)

[Copy](#)

[Close](#)

The following example moves the current selection to the top of the fpList control.

Visual Basic

```
Sub fpList1_SelChange(ItemIndex)
    fpList1.TopIndex = ItemIndex
End Sub
```

See Also

[Accessing List Items](#)

[Searching for List Items](#)

[List](#), [ListCount](#), [ListDown](#), [ListIndex](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

VirtualMode Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether an fpCombo or fpList control displays values from all records or operates in virtual mode.

Syntax

C `UINT LC_GetVirtualMode(HWND hWnd, BOOL FAR *lpValue);`
UINT LC_SetVirtualMode(**HWND** hWnd, **BOOL** value);

C++ `BOOL Class::GetVirtualMode(void);`
Class::SetVirtualMode(**BOOL** value);

Visual Basic `[form.]control.VirtualMode[= boolean%]`

Designer Page

[Virtual Mode designer page](#)

Remarks

The default value for the VirtualMode property is False. When this property is set to False, all the values in the list are displayed.

When the VirtualMode property is set to True, the fpCombo or fpList control only loads the number of rows defined by the [VirtualPageSize](#) property. The remaining rows are retrieved as necessary. As the user scrolls down the list, the fpCombo or fpList control can read records into memory in the background as specified by the VirtualPageSize and [VirtualPagesAhead](#) properties.

When you bind the fpCombo or fpList control to large database tables, reading all the values may affect your application's performance. You can set the VirtualMode property to True to improve speed and performance.

When using virtual mode, the scroll box does not accurately reflect the position of the top item because the control does not know how many records there are. You can set the [VRowCount](#) property to enable the fpCombo or fpList control to create an accurate scroll box range, or you can use the special scroll bars designed for use with virtual mode by setting the [VScrollSpecial](#) and [VScrollSpecialType](#) properties. This scroll bar does not display a thumb print.

Data Type

Integer (Boolean)

See Also

[Using Virtual Mode](#)

[Virtual Mode](#)

[VirtualPagesAhead](#), [VirtualPageSize](#), [VisibleRows](#), [VRowCount](#), [VScrollSpecial](#), [VScrollSpecialType](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

VirtualPagesAhead Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the number of pages the list in an fpCombo or fpList control reads into memory in the background when the [VirtualMode](#) property is set to True.

Syntax

C **UINT** LC_GetVirtualPagesAhead(**HWND** hWnd, **long FAR ***lpValue);
 UINT LC_SetVirtualPagesAhead(**HWND** hWnd, **long** value);

C++ **longClass::**GetVirtualPagesAhead(**void**);
 Class::SetVirtualPagesAhead(**long** value);

Visual Basic [form.]control.VirtualPagesAhead[= value&]

Designer Page

[Virtual Mode designer page](#)

Remarks

As the user scrolls through the list, the number of pages specified by this property are read into memory in the background.

The default value for the VirtualPagesAhead property is 0, which specifies that no pages are read in the background as the user scrolls.

The page size is determined by the [VirtualPageSize](#) property setting.

Data Type

Integer (Long)

See Also

[Using Virtual Mode](#)

[Virtual Mode](#)

[VirtualMode](#), [VirtualPageSize](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

VirtualPageSize Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the size of virtual pages to be read into memory when the [VirtualMode](#) property is set to True.

Syntax

C **UINT** LC_GetVirtualPageSize(**HWND** hWnd, **long FAR** *lpValue);
UINT LC_SetVirtualPageSize(**HWND** hWnd, **long** value);

C++ **longClass::**GetVirtualPageSize(**void**);
Class::SetVirtualPageSize(**long** value);

Visual Basic [form.]control.VirtualPageSize[= value&]

Designer Page

[Virtual Mode designer page](#)

Remarks

The list in an fpCombo or fpList control reads memory one page at a time. The list then displays as many values as possible given its size on the screen. Additional values are read from memory until the list reaches the end of the data currently in memory, at which time an additional page is read.

Pages consist of rows of data to be displayed in the list. Pages can contain the same number of records as can be displayed in the list, or they can contain more records to speed data retrieval.

The page size is the number of rows of data that are read into memory at one time. The default page size is the number of rows that will fit in the displayed list, but you can specify a larger page size so that more rows are read at a time. The value of the VirtualPageSize property indicates the number of rows to be read at a time.

The default value for the VirtualPageSize property is 0, which sets the page size as the current value of the [VisibleRows](#) property. If you set this property to a value less than the value of the VisibleRows property, the virtual page size will be set to the value of the VisibleRows property. For example, if the value of the VisibleRows property is 10 and you set the VirtualPageSize property to 8, the virtual page size will be 10 rows.

You can load additional pages into memory in the background as the user works by setting the [VirtualPagesAhead](#) property.

Data Type

Integer (Long)

See Also

[Using Virtual Mode](#)

[Virtual Mode](#)

[VirtualMode](#), [VirtualPagesAhead](#), [VisibleRows](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

VisibleRows Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Returns the number of rows currently displayed in the list in an fpCombo or fpList control.

Syntax

C **UINT** LC_GetVisibleRows(**HWND** hWnd, **long FAR ***lpValue);

C++ **long** [Class](#)::GetVisibleRows(**void**);

Visual Basic [*form.*]control.VisibleRows[= *value*&]

Remarks

To set the number of rows displayed in the drop-down list of an fpCombo control, use the [MaxDrop](#) property.

Data Type

Integer (Long)

See Also

[MaxDrop](#), [VirtualMode](#), [VirtualPageSize](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

VRowCount Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the scroll box range for an fpCombo or fpList control operating in virtual mode.

Syntax

C **UINT** LC_GetVRowCount(**HWND** hWnd, **long FAR ***lpValue);
UINT LC_SetVRowCount(**HWND** hWnd, **long** value);

C++ **longClass::**GetVRowCount(**void**);
Class::SetVRowCount(**long** value);

Visual Basic [form.]control.VRowCount[= value&]

Designer Page

[Virtual Mode designer page](#)

Remarks

Setting the VRowCount property to the total number of list items enables the fpCombo or fpList control to calculate an accurate scroll box range during virtual mode when using the default scroll bars. By setting the VRowCount property, the scroll box accurately reflects the position of the current selection.

The default value for the VRowCount property is 0, which internally defaults to 5,000 list items.

When the [VirtualMode](#) property is set to True, the fpCombo or fpList control is in virtual mode and reads only a specified number of list items at a time. However, the fpCombo or fpList control cannot calculate an accurate scroll box range because it does not know how many items are in the list. The VRowCount property saves time by providing this value.

The scroll box position will be most accurate if you set the VRowCount property to the exact number of items in the list. If you do not know the exact number of list items, set the VRowCount property to an approximate number. The approximate number still provides the scroll box with enough information to show the scroll box in more or less the correct position. This approximation is preferable to determining the exact number of items in the list because it can be very time consuming to determine the exact number.

Instead of the default scroll bar, you can display a special vertical scroll bar designed to work with virtual mode by setting the [VScrollSpecial](#) property.

Data Type

Integer (Long)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a two-column fpList control that has approximately 4,000 records. Virtual mode is used with a virtual page size of 20. Vertical and horizontal scroll bars are displayed. The scroll box increment is 30 pixels. Special home, end, page up, and page down scroll bars are displayed.

C

```
LC_SetColumns(hWnd, 2);
/* Use virtual mode */
LC_SetVirtualMode(hWnd, TRUE);
/* Display special scroll bars */
LC_SetVScrollSpecial(hWnd, TRUE);
/* Do not display line up and line down arrows */
LC_SetVScrollSpecialType(hWnd, 4);
/* Specify number of database records */
LC_SetVRowCount(hWnd, 4000);
/* Set virtual page size to 20 rows */
LC_SetVirtualPageSize(hWnd, 20);
/* Read ahead 5 pages while user scrolls */
LC_SetVirtualPagesAhead(hWnd, 5);
LC_SetScrollBarH(hWnd, LC_SCROLLBARH_SHOW);
LC_SetScrollBarV(hWnd, LC_SCROLLBARV_SHOW);
/* Set scrolling scale and increment to 30 pixels */
LC_SetScrollHScale(hWnd, LC_SCROLLHSCALE_PIXELS);
LC_SetScrollHInc(hWnd, 30);
```

C++

```
fpList1->SetColumns(2);
// Use virtual mode
fpList1->SetVirtualMode(TRUE);
// Display special scroll bars
fpList1->SetVScrollSpecial(TRUE);
// Do not display line up and line down arrows
fpList1->SetVScrollSpecialType(4);
// Specify number of database records
fpList1->SetVRowCount(4000);
// Set virtual page size to 20 rows
fpList1->SetVirtualPageSize(20);
// Read ahead 5 pages while user scrolls
fpList1->SetVirtualPagesAhead(5);
fpList1->SetScrollBarH(LC_SCROLLBARH_SHOW);
fpList1->SetScrollBarV(LC_SCROLLBARV_SHOW);
// Set scrolling scale and increment to 30 pixels
fpList1->SetScrollHScale(LC_SCROLLHSCALE_PIXELS);
fpList1->SetScrollHInc(30);
```

Visual Basic

```
fpList1.Columns = 2
' Use virtual mode
fpList1.VirtualMode = True
' Display special scroll bars
fpList1.VScrollSpecial = True
' Do not display line up and line down arrows
fpList1.VScrollSpecialType = 4
' Specify number of database records
fpList1.VRowCount = 4000
' Set virtual page size to 20 rows
fpList1.VirtualPageSize = 20
' Read ahead 5 pages while user scrolls
fpList1.VirtualPagesAhead = 5
fpList1.ScrollBarH = LC_SCROLLBARH_SHOW
fpList1.ScrollBarV = LC_SCROLLBARV_SHOW
' Set scrolling scale and increment to 30 pixels
fpList1.ScrollHScale = LC_SCROLLHSCALE_PIXELS
```

```
fpList1.ScrollHInc = 30
```

See Also

[Using Virtual Mode](#)

[Virtual Mode](#)

[VirtualMode](#), [VScrollSpecial](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

VScrollSpecial Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether a customized scroll bar is displayed in an fpCombo or fpList control.

Syntax

C `UINT LC_GetVScrollSpecial(HWND hWnd, BOOL FAR *lpValue);`
UINT LC_SetVScrollSpecial(HWND** hWnd, **BOOL** value);**

C++ `BOOL Class::GetVScrollSpecial(void);`
Class::SetVScrollSpecial(BOOL** value);**

Visual Basic `[form.]control.VScrollSpecial[= boolean%]`

Designer Page

[Virtual Mode designer page](#)

Remarks

The VScrollSpecial property displays a special, customized vertical scroll bar instead of the default Windows scroll bar. Instead of a scroll box, this scroll bar has arrows the user clicks to go to the first or last page, the previous or next page, or the previous or next line. You can remove some of these arrows by setting the [VScrollSpecialType](#) property. The special scroll bar does not display a thumbprint.

The default value for the VScrollSpecial property is False, which means the customized scroll bar is not displayed.

The customized scroll bar works well with an fpCombo or fpList control in virtual mode (when the [VirtualMode](#) property is set to True). In virtual mode, the fpCombo or fpList control reads only as many items as necessary to fill the list.

Although virtual mode improves performance, it prevents the default scroll bar in the fpCombo or fpList control from accurately representing the position of the currently displayed item in the list. You can remedy this problem for the default scroll bar by setting the [VRowCount](#) property, or you can use the customized scroll bar by setting the VScrollSpecial property.

Data Type

Integer (Boolean)

See Also

[Using Virtual Mode](#)

[Virtual Mode](#)

[VirtualMode](#), [VRowCount](#), [VScrollSpecialType](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

VScrollSpecialType Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns which arrows are displayed on the customized scroll bar for an fpCombo or fpList control.

Syntax

C **UINT** LC_GetVScrollSpecialType(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetVScrollSpecialType(**HWND** hWnd, **short** value);

C++ **short** **Class**::GetVScrollSpecialType(**void**);
Class::SetVScrollSpecialType(**short** value);

Visual Basic [form.]control.VScrollSpecialType[= value%]

Designer Page

[Virtual Mode designer page](#)

Remarks

When using a customized scroll bar for an fpCombo or fpList control (by setting the [VScrollSpecial](#) property to True), you can remove some or all of the scroll bar arrows by setting the VScrollSpecialType property.

The following settings can be combined with the Or operator:

Value	Description
0	(Default) Displays all the default arrows on the customized scroll bar
1	Removes the home and end arrows
2	Removes the page up and page down arrows
4	Removes the line up and line down arrows

To remove more than one set of arrows, use the Or operator to combine two or more values. The Or operator lets you perform a bitwise comparison of two numbers. The result sets more than one value. For more information, refer to the Or operator in the Visual Basic documentation.

Data Type

Integer

See Also

[Using Virtual Mode](#)

[Virtual Mode](#)

[VirtualMode](#), [VScrollSpecial](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

WrapList Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether an fpCombo or fpList control wraps data in multiple columns.

Syntax

C `UINT LC_GetWrapList(HWND hWnd, BOOL FAR *pValue);`
`UINT LC_SetWrapList(HWND hWnd, BOOL value);`

C++ `BOOL Class::GetWrapList(void);`
`Class::SetWrapList(BOOL value);`

Visual Basic `[form.]control.WrapList[= boolean%]`

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

The default value for the WrapList property is False.

When the WrapList property is set to False, each column must be set to display data individually using the designated-column properties (such as [Col](#) and [ColWidth](#)).

When the WrapList property is set to True, items from a single-column fpCombo or fpList control are arranged in snaking columns, filling the first column, then the second, and so on. A horizontal scroll bar is displayed if necessary.

When the WrapList property is set to True, all multiple-column support is disabled, and the designated-column property settings are ignored. Also, when the WrapList property is set to True, no vertical scroll bar is displayed regardless of the [ScrollBarV](#) property setting.

Data Type

Integer (Boolean)

See Also

[Wrapping List Items in a Single-Column Control](#)

[Col](#), [Columns](#), [ColWidth](#), [ScrollBarV](#), [WrapWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

WrapWidth Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the width of the columns in an fpCombo or fpList control when the [WrapList](#) property is set to True.

Syntax

C **UINT** LC_GetWrapWidth(*HWND* hWnd, **long FAR ***lpValue);
UINT LC_SetWrapWidth(*HWND* hWnd, **long** value);

C++ **longClass::**GetWrapWidth(**void**);
Class::SetWrapWidth(**long** value);

Visual Basic [form.]control.WrapWidth[= value%]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

When the WrapList property is set to True, list items in the fpCombo or fpList control are displayed in wrapping columns. The WrapWidth property determines the width of the columns.

The default value for the WrapWidth property is 0, which uses the maximum item length.

In Visual Basic, the measurement unit used by the WrapWidth property depends on the setting of the form's ScaleMode property. The default scale mode is twips (1/1440 of an inch). Generally, the ActiveX and VBX controls use twips as the default measurement unit, and the DLL controls use pixels as the default measurement unit.

Data Type

Integer

See Also

[Wrapping List Items in a Single-Column Control](#)

[WrapList](#) property

Properties

[A](#)

[B](#)

[C](#)

[D](#)

[E](#)

[F](#)

[G](#)

[H](#)

[I](#)

[J](#)

[K](#)

[L](#)

[M](#)

[N](#)

[O](#)

[P](#)

[Q](#)

[R](#)

[S](#)

[T](#)

[U](#)

[V](#)

[W](#)

[X](#)

[Y](#)

[Z](#)

[Using the Property Topics](#)

[Understanding Syntax Conventions](#)

[Setting Enumerated Properties](#)

[Loading Pictures in C and C++](#)

[Standard Visual Basic Properties Supported by List Pro Controls](#)

A

[About](#)

[Action](#)

[AlignH](#)

[AlignV](#)

[AllowColDragDrop](#)

[AllowColResize](#)

[AllowGrpDragDrop](#)

[AllowGrpResize](#)

[Appearance](#)

[ApplyTo](#)

[AutoSearch](#)

B

[BackColor](#)

[BorderColor](#)

[BorderDropShadow](#)

[BorderDropShadowColor](#)

[BorderDropShadowWidth](#)

[BorderGrayAreaColor](#)

[BorderStyle](#)

[BorderWidth](#)

C

[Col](#)

[ColDataField](#)

[ColFormat](#)

[ColFromID](#)

[ColFromName](#)

[ColHeaderText](#)

[ColHide](#)

[ColID](#)

[ColLevel](#)

[ColLevelHeight](#)

[ColList](#)

[ColLockResize](#)

[ColMerge](#)

[ColName](#)

[ColParentGroup](#)

[ColPos](#)

[ColPosInParent](#)

[ColsFrozen](#)

[ColSortDataType](#)

[ColSorted](#)

[ColSortSeq](#)

[ColText](#)

[ColumnBound](#)

[ColumnEdit](#)

[ColumnHeaderHeight](#)

[ColumnHeaderShow](#)

[ColumnLevels](#)

[Columns](#)

[ColumnSearch](#)

[ColumnSeparatorChar](#)

[ColumnWidthScale](#)

[ColWidth](#)

[ComboGap](#)

D

[DataAutoHeadings](#)

[DataAutoSizeCols](#)

[DataBookmark](#)

[DataSourcehWnd](#)

[DataSourcehWndList](#)

[DataSourceList](#)

[DataSync](#)

E

[EditHeight](#)

[EnableKeyEvents](#)

[EnableMouseEvents](#)

[EnableTopChangeEvent](#)

[ExtendCol](#)

[ExtendRow](#)

F

[Font](#)

[FontEmpty](#)

[ForeColor](#)

G

[GrayAreaColor](#)

[GroupHeaderHeight](#)

[GroupHeaderShow](#)

[Groups](#)

[Grp](#)

[GrpFromID](#)

[GrpFromName](#)

[GrpHeaderText](#)

[GrpHide](#)

[GrpID](#)

[GrpLockResize](#)

[GrpName](#)

[GrpParentGroup](#)

[GrpPos](#)

[GrpPosInParent](#)

[GrpsFrozen](#)

[GrpWidth](#)

H

[HighestPrecedence](#)

[hWnd](#)

I

[InsertRow](#)

[ItemData](#)

J

[JoinID](#)

L

[Line3DDark](#)

[Line3DLight](#)

[Line3DWidth](#)

[LineApplyTo](#)

[LineColor](#)

[LineStyle](#)

[LineWidth](#)

[List](#)

[List3DText](#)

[List3DTextHighlightColor](#)

[List3DTextOffset](#)

[List3DTextShadowColor](#)

[ListApplyTo](#)

[ListCount](#)

[ListDown](#)

[ListGrayAreaColor](#)

[ListIndex](#)

[ListLeftOffset](#)

[ListWidth](#)

M

[MaxDrop](#)

[MaxEditLen](#)

[MergeAdjustView](#)

[MouseOverArea](#)

[MouseOverCol](#)

[MouseOverColHeader](#)

[MouseOverGrp](#)

[MouseOverGrpHeader](#)

[MouseOverRow](#)

[MultiLine](#)

[MultiSelect](#)

N

[NewIndex](#)

[NextSel](#)

[NoIntegralHeight](#)

P

[Picture](#)

[PictureAlignH](#)

[PictureAlignV](#)

[PictureSel](#)

R

[ReadOnly](#)

[Row](#)

[RowHeight](#)

[RowMerge](#)

S

[ScrollBarH](#)

[ScrollBarV](#)

[ScrollHInc](#)

[ScrollHScale](#)

[SearchIgnoreCase](#)

[SearchIndex](#)

[SearchMethod](#)

[SearchText](#)

[SelCount](#)

[SelDrawFocusRect](#)

[Selected](#)

[SelLength](#)

[SelMax](#)

[SelStart](#)

[SelText](#)

[Sorted](#)

[SortState](#)

[Style](#)

T

[Text](#)

[TextOrientation](#)

[ThreeDFrameColor](#)

[ThreeDFrameWidth](#)

[ThreeDInsideHighlightColor](#)

[ThreeDInsideShadowColor](#)

[ThreeDInsideStyle](#)

[ThreeDInsideWidth](#)

[ThreeDOnFocusInvert](#)

[ThreeDOutsideHighlightColor](#)

[ThreeDOutsideShadowColor](#)

[ThreeDOutsideStyle](#)

[ThreeDOutsideWidth](#)

[TopIndex](#)

V

[VirtualMode](#)

[VirtualPagesAhead](#)

[VirtualPageSize](#)

[VisibleRows](#)

[VRowCount](#)

[VScrollSpecial](#)

[VScrollSpecialType](#)

W

[WrapList](#)

[WrapWidth](#)

This function is available for these classes:

CfpComboBox CfpListBox

No properties begin with this letter.

Using the Property Topics

The following table describes each section in the property topics.

Topic Section	Description																																													
Summary Box	<p>Illustrates</p> <table border="1"> <thead> <tr> <th>PD</th> <th>RD</th> <th>WR</th> <th>RT</th> <th>DT</th> </tr> </thead> <tbody> <tr> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> </tr> </tbody> </table> <p>Whether the property can be set using the FarPoint Property Designer</p> <table border="1"> <thead> <tr> <th>PD</th> <th>RD</th> <th>WR</th> <th>RT</th> <th>DT</th> </tr> </thead> <tbody> <tr> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> </tr> </tbody> </table> <p>Whether the property is read/write, read-only, or write-only</p> <table border="1"> <thead> <tr> <th>PD</th> <th>RD</th> <th>WR</th> <th>RT</th> <th>DT</th> </tr> </thead> <tbody> <tr> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> </tr> </tbody> </table> <p>Whether the property is available at run time, design time, or both</p> <table border="1"> <thead> <tr> <th>Property Designer</th> <th>Read</th> <th>Write</th> <th>Run Time</th> <th>Design Time</th> </tr> <tr> <th>PD</th> <th>RD</th> <th>WR</th> <th>RT</th> <th>DT</th> </tr> </thead> <tbody> <tr> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> </tr> </tbody> </table>	PD	RD	WR	RT	DT	✓	✓	✓	✓	✓	PD	RD	WR	RT	DT	✓	✓	✓	✓	✓	PD	RD	WR	RT	DT	✓	✓	✓	✓	✓	Property Designer	Read	Write	Run Time	Design Time	PD	RD	WR	RT	DT	✓	✓	✓	✓	✓
PD	RD	WR	RT	DT																																										
✓	✓	✓	✓	✓																																										
PD	RD	WR	RT	DT																																										
✓	✓	✓	✓	✓																																										
PD	RD	WR	RT	DT																																										
✓	✓	✓	✓	✓																																										
Property Designer	Read	Write	Run Time	Design Time																																										
PD	RD	WR	RT	DT																																										
✓	✓	✓	✓	✓																																										
Applies To	Lists which List Pro controls support the property																																													
Description	Provides a brief summary of the property																																													
Syntax	Shows how to set or read the value of the property in code (See Understanding Syntax Conventions for information about typographic usage in syntax statements.)																																													
	<p>Notes</p> <table border="1"> <thead> <tr> <th>PD</th> <th>RD</th> <th>WR</th> <th>RT</th> <th>DT</th> </tr> </thead> <tbody> <tr> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> </tr> </tbody> </table> <p>For C and C++ development environments, the "syntax" statements are actually define statements, provided to give you the most information about the List Pro functions.</p> <table border="1"> <thead> <tr> <th>PD</th> <th>RD</th> <th>WR</th> <th>RT</th> <th>DT</th> </tr> </thead> <tbody> <tr> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> </tr> </tbody> </table> <p>Unless stated otherwise, C++ syntax represents MFC syntax for the DLL controls.</p>	PD	RD	WR	RT	DT	✓	✓	✓	✓	✓	PD	RD	WR	RT	DT	✓	✓	✓	✓	✓																									
PD	RD	WR	RT	DT																																										
✓	✓	✓	✓	✓																																										
PD	RD	WR	RT	DT																																										
✓	✓	✓	✓	✓																																										
ActiveX Use	Provides ActiveX information about the property																																													
Designer Page	Lists the FarPoint Property Designer Page and subtab corresponding to the property, if applicable																																													
Remarks	Provides detailed and related information, including enumerated property settings and constants (See Setting Enumerated Properties for more information about enumerated property settings.)																																													
Data Type	Indicates the Visual Basic data type of the property's values																																													
See Also	Provides references to related information																																													
Example	Provides C, C++, and Visual Basic example code that you can print or copy to place in your project																																													
	<p>Note Properties that are not set in the examples use their default values. To determine the default value for a property, refer to its description.</p>																																													

Understanding Syntax Conventions

The "Syntax" section uses the typographic conventions listed in the following table.

Example	Description
<i>value, color, control</i>	Italicized items are placeholders for information you supply. (Substitute the name of the control for the word <i>control</i> in syntax statements.)
short, BOOL	Bold, italicized arguments indicate data types, pointers, and

Class

user-defined types.

In programming syntax, blue text indicates additional information is provided in a pop-up window. To access the pop-up window, click the blue text. (Italicized blue text indicates variable content, as described in the pop-up window.)

[= *value&*]

Items inside square brackets are optional.

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

For C and C++ development environments, the "syntax" statements are actually define statements provided to give you the most information about the List Pro functions.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Unless otherwise stated, C++ syntax represents MFC syntax for the DLL controls.

Setting Enumerated Properties

Enumerated properties are those with a finite number of settings that can be specified with whole numbers (integers) or constants. For example, the fpCombo control's [AlignH](#) property can be set to Default (0 or LC_ALIGNH_DEFAULT), Left (1 or LC_ALIGNH_LEFT), Center (2 or LC_ALIGNH_CENTER), or Right (3 or LC_ALIGNH_RIGHT).

When setting enumerated properties in code, you can use either of the following methods:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Set the property to an integer value, for example,

```
fpCombo1.AlignH = 2
```

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Set the property to a constant, for example,

```
fpCombo1.AlignH = LC_ALIGNH_CENTER
```

Using constants makes your code easier to read and requires fewer comments. In addition, constants are more readily recognized by later versions of the controls because they do not change as integer values sometimes do.

Constants are defined for all List Pro enumerated properties in constants files. The Setup program copies constants files into the `\LSTPRO20\INCLUDE\` subdirectory. To use constants in your code, you must include the constants files in your applications.

Loading Pictures in C and C++

If you are using the C or C++ programming language and are loading pictures in your control, you must use additional code before setting picture properties.

If you are using the C programming language, use code similar to the following to load a picture.

```
HBITMAP hbmpEducation;
HICON hiconHome;
hiconHome = LoadIcon(hInstance, "Home");
hbmpEducation = LoadBitmap(hInstance, "Education");

/* Load bitmap in the first column */
LC_SetCol(hWnd, 1);
LC_SetRow(hWnd, 1);
LC_SetPicture(hWnd, FP_PICT_TYPE_BITMAP, FP_HNDTYPE_HANDLE, hbmpEducation, 0);

/* Load icon in the first column */
LC_SetCol(hWnd, 1);
LC_SetRow(hWnd, 1);
LC_SetPicture(hWnd, FP_PICT_TYPE_ICON, FP_HNDTYPE_HANDLE, hiconHome, 0);
```

If you are using the C++ programming language and a List Pro DLL control, use code similar to the following to load a picture (the bitmap is created in AppStudio).

```
/* using the resource ID with LC class calls */
m_comboControl.SetPicture(FP_PICT_TYPE_BITMAP, FP_HNDTYPE_RESID, IDC_LIST, 0);
```

If you are using the C++ programming language and a List Pro ActiveX control, you must use additional code before setting picture properties. Use the following instructions to load the picture.

1. Declare a picture holder class.

```
CPictureHolder myPicture;
```

2. Create the picture using the Create function.

```
// Create picture from icon
myPicture.CreateFromIcon(IDI_MYICON);
```

3. Provide the picture for the control.

```
m_comboControl.SetPicture(myPicture.GetPictureDispatch( ));
```

If you are using the C++ programming language and a List Pro VBX control, use code similar to the following to convert an hBitmap into an hPic (the bitmap is created in AppStudio).

```
// Convert hBitmap to hPic for the VBX using C++
PIC picture;
HPIC hPic;
HBITMAP hBitmap;

hBitmap=LoadBitmap(AfxGetInstanceHandle( ), "bitmap");
picture.picData.bmp.hbitmap=hBitmap;
picture.picType=PICTYPE_BITMAP;
hPic=AfxSetPict(NULL, &picture);
fpList1->SetPicture(hPic);
```

Standard Visual Basic Properties Supported by List Pro Controls

The following table lists the standard Visual Basic properties that are supported by List Pro controls.

Refer to the Visual Basic documentation for more information on these properties.

DataChanged	HelpContextID
DataField	Index
DataSource	Left
DragIcon	MouseIcon (32-bit VB4 only)
DragMode	MousePointer
Enabled	Name
FontBold	Parent
FontItalic	TabIndex
FontName	TabStop
FontSize	Tag
FontStrikethru	Top
FontUnderline	Visible
Height	Width

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

About Property

[Example](#)

Applies To

fpCombo, fpList controls

Description

Returns version information at run time. At design time, double-clicking this property displays a dialog box with version information. This property is available for VBX controls only.

Syntax

C++ **LPSTR Class**::GetAbout(**LPSTR** buffer, **UINT** nBufferSize);
Visual Basic [form.]control.About

Remarks

The About dialog box displays the List Pro version number, control name, and version date. At run time, you can retrieve the version number as a string and can convert the string to a floating-point number. The version number format is "V.R.NNN", where "V" is the one-digit version number from 0 through 9, "R" is the one-digit update or revision number from 0 through 9, and "NNN" is the three-digit update or maintenance release number from 000 through 999.

Data Type

String

[Print](#)

[Copy](#)

[Close](#)

The following example checks the version number of an fpList control and exits if the version is less than 1.5.000.

C++

```
// If the version number is less than 1.5.000, display a message box and exit
If (m_fpList1->GetAboutBox( ) < "1.5.000")
{
    MessageBox("A later version of List Pro is needed to run this application.");
}
Else
{
    MessageBox("This is a current VBX.");
}
```

Visual Basic

```
' If the version number is less than 1.5.000, display a message box and exit
If fpList1.About <= "1.5.000" Then
    Beep
    MsgBox "A later version of List Pro is needed to run this application."
End
Else
    MsgBox "This is a current VBX."
End If
```

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Action Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets a value that designates an action, such as inserting a column or deleting a row. This property is available at run time only.

Syntax

C **UINT** LC_SetAction(**HWND** hWnd, **short** value);

C++ **Class::**SetAction(**short** value);

Visual Basic [form.]control.Action[= setting%]

Designer Page

Setting 7 (Insert Column) and setting 8 (Delete Column) correspond to the Insert Column and Delete Column buttons on the [Specific subtab of the Columns designer page](#). Setting 10 (Insert Group) and setting 11 (Delete Group) correspond to the Insert Group and Delete Group buttons on the [Specific subtab of the Groups designer page](#). Setting 12 (Clone Column) corresponds to the Clone Column button on the Specific subtab of the Columns designer page. Setting 4 (Delete Row) corresponds to the Delete Row button on the [Add Data designer page](#).

Remarks

Use the Action property in place of Visual Basic methods for the fpCombo and fpList controls. For example, use setting 4 (Delete Row) to delete the one hundred thousandth list item in a database because the Visual Basic RemoveItem method supports only up to 32,768 list items.

The following settings are available:

Setting	Description	Constant
0 - Search	Performs a search on the list based on the search properties, such as SearchIgnoreCase , SearchText , and SearchMethod	LC_ACTION_SEARCH
1 - Select All	Selects all rows in the list box when the MultiSelect property is set to 1 (Simple) or 2 (Extended)	LC_ACTION_SELECTALL
2 - Deselect All	Deselects all items or rows in the list	LC_ACTION_DESELECTALL
3 - Clear	Clears all items or rows from the list	LC_ACTION_CLEAR
4 - Delete Row	Deletes an item or row (Use the Row property to specify the row to delete.)	LC_ACTION_DELETEROW
5 - Force Update	Forces the list box to display changes	LC_ACTION_FORCEUPDATE
6 - Clear Search Buffer	Clears the search buffer when users search using multiple characters by way of the AutoSearch property	LC_ACTION_CLEARSEARCHBUFFER
7 - Insert Column	Inserts a column before the specified column (Use the Col property to specify where to insert the new column.)	LC_ACTION_INSERTCOL
8 - Delete Column	Deletes a column (Use the Col property to specify the column to	LC_ACTION_DELETECOL

9 - Virtual Refresh	delete.) Forces the control to discard the current page of data and re-request the data currently in the buffer (Use this setting with the VirtualMode property.)	LC_ACTION_VIRTUALREFRESH
10 - Insert Group	Inserts a group before the specified group (Use the Grp property to specify where to insert the new group.)	LC_ACTION_INSERTGRP
11 - Delete Group	Deletes a group (Use the Grp property to specify the group to delete.)	LC_ACTION_DELETEGRP
12 - Clone Column	Copies a column (all attributes except data, column identification number or name, and data field setting) and inserts the cloned column to the right of the copied column (Use the Col property to specify which column to clone.)	LC_ACTION_CLONECOL

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example adds rows to an fpList control, and then allows the user to delete the currently selected row.

C

```
LC_SetRow(hWnd, -1);
LC_SetInsertRow(hWnd, "North Carolina");
LC_SetInsertRow(hWnd, "Ohio");
LC_SetInsertRow(hWnd, "Virginia");
LC_SetInsertRow(hWnd, "Florida");
LC_SetInsertRow(hWnd, "California");

void OnLButtonDbClickLB(UINT, int, Cwnd*, LPVOID)
{
    long Index;
    LC_SetRow(hWnd, LC_GetListIndex(hWnd));
    LC_SetRow(hWnd, LC_GetListIndex(hWnd, &Index));
    LC_SetAction(hWnd, LC_ACTION_DELETEROW);
}
```

C++

```
bool CListBox_Dlg::OnInitDialog( )
{
    m_LB->SetRow(-1);
    m_LB->InsertRow("North Carolina");
    m_LB->InsertRow("Ohio");
    m_LB->InsertRow("Virginia");
    m_LB->InsertRow("Florida");
    m_LB->InsertRow("California");
}

void CListBox_Dlg::OnLButtonDbClick(UINT, int, Cwnd*, LPVOID lpparams)
{
    m_LB->SetRow(m_LB->GetListIndex( ));
    m_LB->SetAction(LC_ACTION_DELETEROW);
}
```

Visual Basic

```
Sub Form_Load ( )
fpList1.Row = -1
fpList1.InsertRow = "North Carolina"
fpList1.InsertRow = "Ohio"
fpList1.InsertRow = "Virginia"
fpList1.InsertRow = "Florida"
fpList1.InsertRow = "California"
End Sub

Sub fpList1_DblClick ( )
Dim Choice As Integer

' Prompt if the user wants to delete the double-clicked item
Choice = MsgBox("Are you sure you want to delete this item?", 4)

' If Yes
If Choice = 6 Then
    ' Set current row
    fpList1.Row = fpList1.ListIndex
    ' Delete item
    fpList1.Action = LC_ACTION_DELETEROW
Else
    Exit Sub
End If
End Sub
```

See Also

[Cloning Columns](#)

[Creating Multiple Columns](#)

[Creating Groups](#)

[Searching for List Items](#)

[Sorting List Items](#)

[AutoSearch](#), [Col](#), [Grp](#), [MultiSelect](#), [Row](#), [SearchIgnoreCase](#), [SearchIndex](#), [SearchMethod](#), [SearchText](#), [VirtualMode](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

AlignH Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the horizontal alignment of text in a column or group.

Syntax

C *UINT* LC_GetAlignH(*HWND* hWnd, *short FAR* *pValue);
UINT LC_SetAlignH(*HWND* hWnd, *short* value);

C++ *short Class*::GetAlignH(*void*);
Class::SetAlignH(*short* value);

Visual Basic [form.]control.AlignH[= setting%]

Designer Page

[List subtab of the ApplyTo designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Default	Left justifies the text in cells Centers the text in headers	LC_ALIGNH_DEFAULT
1 - Left	(Default) Left-justifies the text	LC_ALIGNH_LEFT
2 - Center	Centers the text	LC_ALIGNH_CENTER
3 - Right	Right-justifies the text	LC_ALIGNH_RIGHT

You can set the [ListApplyTo](#) property before you set the AlignH property to designate the part of the control to which text alignment applies.

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example horizontally centers the column header text and vertically aligns text at the bottom of all rows.

C

```
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_COL_HEADER);
LC_SetAlignH(hWnd, LC_ALIGNH_CENTER);
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_ALL_ROWS);
LC_SetAlignV(hWnd, LC_ALIGNV_BOTTOM);
```

C++

```
fpList1->SetListApplyTo(LC_LISTAPPLYTO_COL_HEADER);
fpList1->SetAlignH(LC_ALIGNH_CENTER);
fpList1->SetListApplyTo(LC_LISTAPPLYTO_ALL_ROWS);
fpList1->SetAlignV(LC_ALIGNV_BOTTOM);
```

Visual Basic

```
fpList1.ListApplyTo = LC_LISTAPPLYTO_COL_HEADER
fpList1.AlignH = LC_ALIGNH_CENTER
fpList1.ListApplyTo = LC_LISTAPPLYTO_ALL_ROWS
fpList1.AlignV = LC_ALIGNV_BOTTOM
```

See Also

[Aligning Text and Graphics](#)

[AlignV](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

AlignV Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the vertical alignment of text in a column or group.

Syntax

C *UINT* LC_GetAlignV(*HWND* hWnd, *short FAR* *lpValue);
 UINT LC_SetAlignV(*HWND* hWnd, *short* value);

C++ *short* *Class*::GetAlignV(*void*);
 Class::SetAlignV(*short* value);

Visual Basic [form.]control.AlignV[= setting%]

Designer Page

[List subtab of the ApplyTo designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Default	Centers the text vertically	LC_ALIGNV_DEFAULT
1 - Top	Displays the text at the top	LC_ALIGNV_TOP
2 - Center	(Default) Centers the text vertically	LC_ALIGNV_CENTER
3 - Bottom	Displays the text at the bottom of the row	LC_ALIGNV_BOTTOM

You can set the [ListApplyTo](#) property before you set the AlignV property to designate the part of the control to which text alignment applies.

Data Type

Integer (Enumerated)

See Also

[Aligning Text and Graphics](#)

[AlignH](#), [ListApplyTo](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

AllowColDragDrop Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether the user can move columns at run time in a multiple-column fpCombo or fpList control by dragging and dropping a column header.

Syntax

C **UINT** LC_GetAllowColDragDrop(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetAllowColDragDrop(**HWND** hWnd, **short** value);

C++ **short Class::**GetAllowColDragDrop(**void**);
Class::SetAllowColDragDrop(**short** value);

Visual Basic [form.]control.AllowColDragDrop[= setting%]

Designer Page

[General subtab of the Columns designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Off	(Default) Prevents columns from being dragged and dropped	LC_ALLOWCOLDRAGDROP_OFF
1 - All Cols	Enables any column to be dragged and dropped	LC_ALLOWCOLDRAGDROP_ALLCOLS
2 - Non Frozen Cols	Enables any column except frozen columns to be dragged and dropped	LC_ALLOWCOLDRAGDROP_NONFROZENCOLS

When the AllowColDragDrop property is set to 1 (All Cols) or 2 (Non Frozen Cols), users can click the header of a column they want to move, drag it to a new position, and release the mouse button to move the column.

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If you move a column, the index number of any column affected by the move will change. When you move a column to the left of its current position, the column index number of the column you move and of all columns to the right of the destination column will change. When you move a column to the right of its current position, the column index number of the column you move and of all columns between it and the destination column will change. For more information, see [Referencing a Column](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

You must display column headers to drag and drop columns.

To create a multiple-column fpCombo or fpList control, set the [Columns](#) property to a value greater than 0.

To prevent frozen columns from being moved, set this property to 2 (Non Frozen Cols).

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a four-column fpList control. The first column is frozen, but the user can rearrange or resize other columns by dragging and dropping at run time.

C

```
LC_SetColumns(hWnd, 4);
LC_SetAllowColDragDrop(hWnd, LC_ALLOWCOLDRAGDROP_NONFROZENCOLS);
/* Freeze first column */
LC_SetColsFrozen(hWnd, 1);
LC_SetAllowColResize(hWnd, LC_ALLOWCOLRESIZE_RESIZECOLORHEADER);
/* Show header */
LC_SetColumnHeaderShow(hWnd, TRUE);
/* First column */
LC_SetCol(hWnd, 0);
LC_SetColHeaderText(hWnd, "Name");
/* Second column */
LC_SetCol(hWnd, 1);
LC_SetColHeaderText(hWnd, "PubID");
/* Third column */
LC_SetCol(hWnd, 2);
LC_SetColHeaderText(hWnd, "Telephone");
/* Fourth column */
LC_SetCol(hWnd, 3);
LC_SetColHeaderText(hWnd, "Fax");
```

C++

```
fpList1->SetColumns(4);
fpList1->SetAllowColDragDrop(LC_ALLOWCOLDRAGDROP_NONFROZENCOLS);
// Freeze first column
fpList1->SetColsFrozen(1);
fpList1->SetAllowColResize(LC_ALLOWCOLRESIZE_RESIZECOLORHEADER);
// Show header
fpList1->SetColumnHeaderShow(TRUE);
// First column
fpList1->SetCol(0);
fpList1->SetColHeaderText("Name");
// Second column
fpList1->SetCol(1);
fpList1->SetColHeaderText("PubID");
// Third column
fpList1->SetCol(2);
fpList1->SetColHeaderText("Telephone");
// Fourth column
fpList1->SetCol(3);
fpList1->SetColHeaderText("Fax");
```

Visual Basic

```
fpList1.Columns = 4
fpList1.AllowColDragDrop = LC_ALLOWCOLDRAGDROP_NONFROZENCOLS
' Freeze first column
fpList1.ColsFrozen = 1
fpList1.AllowColResize = LC_ALLOWCOLRESIZE_RESIZECOLORHEADER
' Show header
fpList1.ColumnHeaderShow = True
' First column
fpList1.Col = 0
fpList1.ColHeaderText = "Name"
' Second column
fpList1.Col = 1
fpList1.ColHeaderText = "PubID"
' Third column
fpList1.Col = 2
fpList1.ColHeaderText = "Telephone"
```

```
' Fourth column  
fpList1.Col = 3  
fpList1.ColHeaderText = "Fax"
```


See Also

[Referencing a Column](#)

[Moving Columns in the Control](#)

[ColsFrozen](#), [Columns](#) properties

[DragDropCol](#) event

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

AllowColResize Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether the user can resize columns in a multiple-column fpCombo or fpList control.

Syntax

C *UINT* LC_GetAllowColResize(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetAllowColResize(**HWND** hWnd, **short** value);

C++ **short** *Class*::GetAllowColResize(**void**);
Class::SetAllowColResize(**short** value);

Visual Basic [form.]control.AllowColResize[= setting%]

Designer Page

[General subtab of the Columns designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Off	(Default) Prevents the user from resizing the column by dragging and dropping a boundary	LC_ALLOWCOLRESIZE_OFF
1 - Resize Header	Enables the user to resize the column by dragging a header boundary	LC_ALLOWCOLRESIZE_RESIZEHEADER
2 - Resize Col or Header	Enables the user to resize the column by dragging either a header boundary or a column boundary	LC_ALLOWCOLRESIZE_RESIZECOLORHEADER

When the AllowColResize property is set to a value other than 0 (Off), the user can resize columns in a multiple-column fpCombo or fpList control by dragging a column boundary. When the mouse passes over a column boundary, the pointer changes to a resize pointer. The user can press the left mouse button and drag the boundary to resize the column.

Setting the [ColLockResize](#) property to a column number prevents a specific column from being resized.

When the user releases the mouse button after resizing the column, the [ColWidthChange](#) event occurs.

To create a multiple-column fpCombo or fpList control, set the [Columns](#) property to a value greater than 0.

Data Type

Integer (Enumerated)

See Also

[Resizing Columns](#)

[ColLockResize](#), [Columns](#) properties

[ColWidthChange](#) event

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

AllowGrpDragDrop Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether the user can move groups at run time in a multiple-group fpCombo or fpList control by dragging and dropping a group header.

Syntax

C **UINT** LC_GetAllowGrpDragDrop(**HWND** hWnd, **short FAR** *pValue);
UINT LC_SetAllowGrpDragDrop(**HWND** hWnd, **short** value);

C++ **short Class::**GetAllowGrpDragDrop(**void**);
Class::SetAllowGrpDragDrop(**short** value);

Visual Basic [form.]control.AllowGrpDragDrop[= setting%]

Designer Page

[General subtab of the Groups designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Off	(Default) Prevents groups from being dragged and dropped	LC_ALLOWGRPDRAHDROP_OFF
1 - All Grps	Enables any group to be dragged and dropped	LC_ALLOWGRPDRAHDROP_ALLGRPS
2 - Non Frozen Grps	Enables any group except frozen groups to be dragged and dropped	LC_ALLOWGRPDRAHDROP_NONFROZENGRPS

When the AllowGrpDragDrop property is set to 1 (All Grps) or 2 (Non Frozen Grps), users can click the header of a group they want to move, drag it to a new position, and release the mouse button to move the group.

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If you move a group, the index number of any group affected by the move will change. When you move a group to the left of its current position, the group index number of the group you move and of all groups to the right of the destination group will change. When you move a group to the right of its current position, the group index number of the group you move and of all groups between it and the destination group will change. For more information, see [Referencing a Group](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

You must display group headers to drag and drop groups.

To create a multiple-group fpCombo or fpList control, set the [Groups](#) property to a value greater than 0.

To prevent frozen groups from being moved, set this property to 2 (Non Frozen Grps).

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a list box control with two groups and four columns. The user can drag and drop all nonfrozen groups and can resize a group by dragging a header or group boundary.

C

```
LC_SetColumns(hWnd, 4);
LC_SetGroups(hWnd, 2);
LC_SetGroupHeaderShow(hWnd, TRUE);
/* Allow group resize */
LC_SetAllowGrpResize(hWnd, LC_ALLOWGRPRESIZE_RESIZEGRPORHEADER);
/* Allow group drag and drop */
LC_SetAllowGrpDragDrop(hWnd, LC_ALLOWGRPDRAGDROP_NONFROZENGRPS);
/* Create 1st group */
LC_SetGrp(hWnd, 0);
LC_SetGrpHeaderText(hWnd, "SSN");
/* Create 2nd group */
LC_SetGrp(hWnd, 1);
LC_SetGrpHeaderText(hWnd, "Date of Birth");
/* Assign columns to groups */
LC_SetCol(hWnd, 0);
LC_SetColParentGroup(hWnd, 0);
LC_SetCol(hWnd, 1);
LC_SetColParentGroup(hWnd, 0);
LC_SetCol(hWnd, 2);
LC_SetColParentGroup(hWnd, 1);
LC_SetCol(hWnd, 3);
LC_SetColParentGroup(hWnd, 1);
```

C++

```
fpList1->SetColumns(4);
fpList1->SetGroups(2);
fpList1->SetGroupHeaderShow(TRUE);
// Allow group resize
fpList1->SetAllowGrpResize(LC_ALLOWGRPRESIZE_RESIZEGRPORHEADER);
// Allow group drag and drop
fpList1->SetAllowGrpDragDrop(LC_ALLOWGRPDRAGDROP_NONFROZENGRPS);
// Create 1st group
fpList1->SetGrp(0);
fpList1->SetGrpHeaderText("SSN");
// Create 2nd group
fpList1->SetGrp(1);
fpList1->SetGrpHeaderText("Date of Birth");
// Assign columns to groups
fpList1->SetCol(0);
fpList1->SetColParentGroup(0);
fpList1->SetCol(1);
fpList1->SetColParentGroup(0);
fpList1->SetCol(2);
fpList1->SetColParentGroup(1);
fpList1->SetCol(3);
fpList1->SetColParentGroup(1);
```

Visual Basic

```
fpList1.Columns = 4
fpList1.Groups = 2
fpList1.GroupHeaderShow = True
' Allow group resize
fpList1.AllowGrpResize = LC_ALLOWGRPRESIZE_RESIZEGRPORHEADER
' Allow group drag and drop
fpList1.AllowGrpDragDrop = LC_ALLOWGRPDRAGDROP_NONFROZENGRPS
' Create 1st group
fpList1.Grp = 0
fpList1.GrpHeaderText = "SSN"
```

```
' Create 2nd group
fpList1.Grp = 1
fpList1.GrpHeaderText = "Date of Birth"
' Assign columns to groups
fpList1.Col = 0
fpList1.ColParentGroup = 0
fpList1.Col = 1
fpList1.ColParentGroup = 0
fpList1.Col = 2
fpList1.ColParentGroup = 1
fpList1.Col = 3
fpList1.ColParentGroup = 1
```

See Also

[Referencing a Group](#)

[Moving Groups in the Control](#)

[Groups](#), [Grp](#), [GrpsFrozen](#) properties

[DragDropGrp](#) event

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

AllowGrpResize Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether the user can resize groups in a multiple-group fpCombo or fpList control.

Syntax

C **UINT** LC_GetAllowGrpResize(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetAllowGrpResize(**HWND** hWnd, **short** value);

C++ **short Class::**GetAllowGrpResize(**void**);
Class::SetAllowGrpResize(**short** value);

Visual Basic [form.]control.AllowGrpResize[= setting%]

Designer Page

[General subtab of the Groups designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Off	(Default) Prevents the user from resizing the group by dragging and dropping a boundary	LC_ALLOWGRPRESIZE_OFF
1 - Resize Header	Enables the user to resize the group by dragging a header boundary	LC_ALLOWGRPRESIZE_RESIZEHEADER
2 - Resize Grp or Header	Enables the user to resize the group by dragging either a header boundary or a group boundary	LC_ALLOWGRPRESIZE_RESIZEGRPORHEADER

When the AllowGrpResize property is set to a value other than 0 (Off), the user can resize groups in a multiple-group fpCombo or fpList control by dragging a group boundary. When the mouse passes over a group boundary, the pointer changes to a resize pointer. The user can press the left mouse button and drag the boundary to resize the group.

Setting the [GrpLockResize](#) property to a group number prevents a specific group from being resized.

When the user releases the mouse button after resizing the group, the [GrpWidthChange](#) event occurs.

To create a multiple-group fpCombo or fpList control, set the [Groups](#) property to a value greater than 0.

Data Type

Integer (Enumerated)

See Also

[Resizing Groups](#)

[Groups](#), [GrpLockResize](#) properties

[GrpWidthChange](#) event

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Appearance Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns predefined border styles.

Syntax

C **UINT** LC_GetAppearance(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetAppearance(**HWND** hWnd, **short** value);

C++ **short** **Class**::GetAppearance(**void**);
Class::SetAppearance(**short** value);

Visual Basic [form.]control.Appearance[= setting%]

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Custom	Uses the current border settings	LC_APPEARANCE_CUSTOM
1 - Flat	(Default) Creates a control with an outline border and no inner or outer three-dimensional borders, similar to Windows 3.1 and Windows NT 3.5	LC_APPEARANCE_FLAT
2 - 3-D	Creates a control with inner and outer three-dimensional borders and no outline border, similar to Windows 95	LC_APPEARANCE_3D
3 - 3-D with Border	Creates a control with an outline border and inner and outer three-dimensional borders	LC_APPEARANCE_3D_W_BORDER

The Appearance property settings correspond to predefined settings of the following properties:

BorderColor	ThreeDInsideHighlightColor	ThreeDOutsideHighlightColor
BorderStyle	ThreeDInsideShadowColor	ThreeDOutsideShadowColor
BorderWidth	ThreeDInsideStyle	ThreeDOutsideStyle
ThreeDFrameWidth	ThreeDInsideWidth	ThreeDOutsideWidth

If you apply a predefined appearance style, be aware that the settings for these properties might be changed. For complete information, see [Using Predefined Appearance Styles](#).

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a combo box control that displays a three-dimensional appearance with a border. The control also has a 4-pixel three-dimensional blue frame. The gray area color of both the list and the control border is white.

C

```
LC_SetAppearance(hWnd, LC_APPEARANCE_3D_WITH_BORDER);
LC_SetThreeDFrameWidth(hWnd, 4);
/* blue RGB(0, 0, 255) */
LC_SetThreeDFrameColor(hWnd, 0x00FF0000);
/* white RGB(255, 255, 255) */
LC_SetBorderGrayAreaColor(hWnd, 0x00FFFFFF);
/* white RGB(255, 255, 255) */
LC_SetListGrayAreaColor(hWnd, 0x00FFFFFF);
```

C++

```
fpCombo1->SetAppearance(LC_APPEARANCE_3D_WITH_BORDER);
fpCombo1->SetThreeDFrameWidth(4);
// blue RGB(0, 0, 255)
fpCombo1->SetThreeDFrameColor(0x00FF0000);
// white RGB(255, 255, 255)
fpCombo1->SetBorderGrayAreaColor(0x00FFFFFF);
// white RGB(255, 255, 255)
fpCombo1->SetListGrayAreaColor(0x00FFFFFF);
```

Visual Basic

```
fpCombo1.Appearance = LC_APPEARANCE_3D_WITH_BORDER
fpCombo1.ThreeDFrameWidth = 4
' blue RGB(0, 0, 255)
fpCombo1.ThreeDFrameColor = &H00FF0000&
' white RGB(255, 255, 255)
fpCombo1.BorderGrayAreaColor = &H00FFFFFF&
' white RGB(255, 255, 255)
fpCombo1.ListGrayAreaColor = &H00FFFFFF&
```

See Also

[Customizing the Control's Borders](#)

[Using Predefined Appearance Styles](#)

[BorderColor](#), [BorderStyle](#), [BorderWidth](#), [ThreeDFrameWidth](#), [ThreeDInsideHighlightColor](#), [ThreeDInsideShadowColor](#), [ThreeDInsideStyle](#), [ThreeDInsideWidth](#), [ThreeDOutsideHighlightColor](#), [ThreeDOutsideShadowColor](#), [ThreeDOutsideStyle](#), [ThreeDOutsideWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ApplyTo Property

[See Also](#) [Example](#)

Applies To

fpCombo control

Description

Sets or returns which parts of the fpCombo control are affected by the three-dimensional border properties (such as [ThreeDFrameWidth](#)), and foreground and background properties (such as [BackColor](#)). This property is available at run time only.

Syntax

C **UINT** CbxGetApplyTo(**HWND** hWnd, **short FAR** *lpValue);
UINT CbxSetApplyTo(**HWND** hWnd, **short** value);

C++ **short** CfpComboBox::GetApplyTo(**void**);
CfpComboBox::SetApplyTo(**short** value);

Visual Basic [form.]fpCombo1.ApplyTo[= setting%]

Designer Page

[Border subtab of the Appearance designer page](#)
[Color subtab of the Appearance designer page](#)
[Misc subtab of the Appearance designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Both	(Default) Apply properties to both the list and the edit field	CBX_APPLYTO_BOTH
1 - List Only	Apply properties to the list	CBX_APPLYTO_LISTONLY
2 - Edit Only	Apply properties to the edit field	CBX_APPLYTO_EDITONLY

You can use the ApplyTo property before setting any of the following properties:

Action - Clear	BorderWidth	ThreeDInsideShadowColor
Appearance	Font	ThreeDInsideStyle
BackColor	FontEmpty	ThreeDInsideWidth
BorderColor	ForeColor	ThreeDOnFocusInvert
BorderDropShadow	Text	ThreeDOutsideHighlightColor
BorderDropShadowColor	ThreeDFrameColor	ThreeDOutsideShadowColor
BorderDropShadowWidth	ThreeDFrameWidth	ThreeDOutsideStyle
BorderGrayAreaColor	ThreeDInsideHighlightColor	ThreeDOutsideWidth
BorderStyle		

Set the ApplyTo property before you set the affected properties. After you have finished setting the properties for a particular part of the fpCombo control, set the ApplyTo property to 0 (Both) to ensure that succeeding settings apply to both parts.

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example sets different color properties for the edit field and the list in an fpCombo control.

C

```
/* Add items to the fpCombo */
LC_SetRow(hWnd, -1);
LC_SetInsertRow(hWnd, "Row 1");
LC_SetInsertRow(hWnd, "Row 2");
LC_SetInsertRow(hWnd, "Row 3");
/* Set colors for edit field */
CbxSetApplyTo(hWnd, CBX_APPLYTO_EDITONLY);
/* blue RGB(0, 0, 255) */
LC_SetForeColor(hWnd, 0x00FF0000);
/* white RGB(255, 255, 255) */
LC_SetBackColor(hWnd, 0x00FFFFFF);
/* Set colors for list */
CbxSetApplyTo(hWnd, CBX_APPLYTO_LISTONLY);
/* green RGB(0, 255, 0) */
LC_SetForeColor(hWnd, 0x0000FF00);
/* white RGB(255, 255, 255) */
LC_SetBackColor(hWnd, 0x00FFFFFF);
/* Return setting to 0 (Both) */
CbxSetApplyTo(hWnd, CBX_APPLYTO_BOTH);
```

C++

```
// Add items to the fpCombo
fpCombo1->SetRow(-1);
fpCombo1->SetInsertRow("Row 1");
fpCombo1->SetInsertRow("Row 2");
fpCombo1->SetInsertRow("Row 3");
// Set colors for edit field
fpCombo1->SetApplyTo(CBX_APPLYTO_EDIT_ONLY);
// blue RGB(0, 0, 255)
fpCombo1->SetForeColor(0x00FF0000);
// white RGB(255, 255, 255)
fpCombo1->SetBackColor(0x00FFFFFF);
// Set colors for list
fpCombo1->SetApplyTo(CBX_APPLYTO_LIST_ONLY);
// green RGB(0, 255, 0)
fpCombo1->SetForeColor(0x0000FF00);
// white RGB(255, 255, 255)
fpCombo1->SetBackColor(0x00FFFFFF);
// Return setting to 0 (Both)
fpCombo1->SetApplyTo(CBX_APPLYTO_BOTH);
```

Visual Basic

```
' Add items to the fpCombo
fpCombo1.Row = -1
fpCombo1.InsertRow = "Row 1"
fpCombo1.InsertRow = "Row 2"
fpCombo1.InsertRow = "Row 3"
' Set colors for edit field
fpCombo1.ApplyTo = CBX_APPLYTO_EDITONLY
' blue RGB(0, 0, 255)
fpCombo1.ForeColor = &H00FF0000&
' white RGB(255, 255, 255)
fpCombo1.BackColor = &H00FFFFFF&
' Set colors for list
fpCombo1.ApplyTo = CBX_APPLYTO_LISTONLY
' green RGB(0, 255, 0)
fpCombo1.ForeColor = &H0000FF00&
' white RGB(255, 255, 255)
fpCombo1.BackColor = &H00FFFFFF&
```

```
' Return setting to 0 (Both)
fpCombol.ApplyTo = CBX_APPLYTO_BOTH
```

See Also

[Applying Properties to Specific Parts of the Control](#)

[Customizing the Controls Borders](#)

[Action](#), [AlignH](#), [AlignV](#), [Appearance](#), [BackColor](#), [BorderColor](#), [BorderDropShadow](#), [BorderDropShadowColor](#), [BorderDropShadowWidth](#), [BorderGrayAreaColor](#), [BorderWidth](#), [Font](#), [FontEmpty](#), [ForeColor](#), [Text](#), [ThreeDFrameColor](#), [ThreeDFrameWidth](#), [ThreeDInsideHighlightColor](#), [ThreeDInsideShadowColor](#), [ThreeDInsideStyle](#), [ThreeDInsideWidth](#), [ThreeDOnFocusInvert](#), [ThreeDOutsideHighlightColor](#), [ThreeDOutsideShadowColor](#), [ThreeDOutsideStyle](#), [ThreeDOutsideWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

AutoSearch Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the type of search the user can perform when searching for a specific list item.

Syntax

C **UINT** LC_GetAutoSearch(*HWND* hWnd, **short FAR** *pValue);
UINT LC_SetAutoSearch(*HWND* hWnd, **short** value);

C++ **short Class::**GetAutoSearch(**void**);
Class::SetAutoSearch(**short** value);

Visual Basic [form.]control.AutoSearch[= setting%]

Designer Page

[Search designer page](#)

Remarks

The AutoSearch property allows the user to type one or more characters to search for an item in the list of an fpList control or drop-down list style fpCombo control. Typing a character initiates a search for the first item that begins with that character. The control highlights the found item.

If the search character is not found and the setting is 1 (Single Char), no item is highlighted. If the setting is 2 (Multiple Char), the selection highlighting moves to the closest available match for the search string as the characters are entered. If none of the items begin with the first character entered, the fpCombo or fpList control does not move the selection. If the setting is 3 (Single Char (Greater or Equal)), the fpCombo or fpList control highlights the next greater item if the search character is not found. For example, the first item starting with the letter "g" will be highlighted if the user searched for the letter "f" and none of the items begin with that letter.

The following settings are available:

Setting	Description	Constant
0 - None	Removes the searching capability	LC_AUTOSEARCH_NONE
1 - Single Char	(Default) Lets the user type a single search character and highlights the first item containing that character	LC_AUTOSEARCH_SINGLE_CHAR
2 - Multiple Char	Lets the user type more than one search character and highlights the first item containing those characters in sequence (for example, to highlight "Variegated" type "var")	LC_AUTOSEARCH_MULTIPLE_CHAR
3 - Single Char (Greater or Equal)	Lets the user type a single search character and highlights the first item containing the search character or, if the search character is not found, the next item starting with a greater character	LC_AUTOSEARCH_SINGLE_GREATER

Set the [Style](#) property to 2 (Drop-Down List) to create a drop-down list style fpCombo control. Drop-down combo and simple combo style fpCombo controls let users search by typing characters in the edit field. By default, these styles of fpCombo controls always perform a multiple-character search. Therefore, setting the AutoSearch property for drop-down combo or simple combo style fpCombo controls has no effect.

Use the [ColumnSearch](#) property to designate which column to search in a multiple-column fpCombo or fpList control.

Use the [SearchText](#) property to specify a search string at run time. Set the [Action](#) property to 0 (Search) after designating the

search string to perform the search at run time. Set the Action property to 6 (Clear Search Buffer) to clear the buffer after performing a search.

Data Type

Integer (Enumerated)

See Also

[Searching for List Items](#)

[Action](#), [ColumnSearch](#), [SearchMethod](#), [SearchText](#), [Style](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

BackColor, ForeColor Properties

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the background and foreground (text) colors of an fpCombo or fpList control.

Syntax

C `UINT LC_GetBackColor(HWND hWnd, COLORREF FAR *pValue);`
UINT LC_SetBackColor(HWND** hWnd, **COLORREF** value);**

C++ `COLORREF Class::GetBackColor(void);`
Class::SetBackColor(COLORREF** value);**

Visual Basic `[form.]control.BackColor[= color]`

Note The BackColor and ForeColor properties use the same syntax.

Designer Page

[List subtab of the ApplyTo designer page](#)

ActiveX Use

The BackColor and ForeColor properties are stock properties.

Remarks

The default values are &H80000005& (Windows system window color) for the BackColor property and &H80000008& (Windows system window text color) for the ForeColor property.

When using the three-dimensional properties, you can maximize the three-dimensional effect by selecting a foreground color that contrasts with the background color. Also, the control's background color should contrast with the [ThreeDInsideShadowColor](#) and [ThreeDInsideHighlightColor](#) property settings. The form's foreground color should contrast with the [ThreeDOutsideShadowColor](#) and [ThreeDOutsideHighlightColor](#) property settings.

You can set the [ListApplyTo](#) property before you set the BackColor or ForeColor property to designate the part of the control to which the color applies.

Visual Basic users can refer to the Visual Basic documentation for additional information about the BackColor and ForeColor properties.

Data Type

Color

See Also

[Changing the Background Color](#)

[Changing Text Color and Fonts](#)

[ListApplyTo](#), [ThreeDInsideHighlightColor](#), [ThreeDInsideShadowColor](#), [ThreeDOutsideHighlightColor](#), [ThreeDOutsideShadowColor](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

BorderColor Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the color of the outline border.

Syntax

C	UINT LC_GetBorderColor(HWND hWnd, COLORREF FAR *pValue); UINT LC_SetBorderColor(HWND hWnd, COLORREF value);
C++	COLORREF Class ::GetBorderColor(void); Class ::SetBorderColor(COLORREF value);
Visual Basic	[form.]control.BorderColor[= color]

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The default value for the BorderColor property is &H80000006& (Windows system window frame color).

You must choose a solid color for the BorderColor. Also, the [BorderWidth](#) property must be set to a value greater than 0 and the [BorderStyle](#) property must be set to a value other than 0 (No Border) to use the BorderColor property.

Every control has an outline border. In addition, the control can display inner and outer borders that can create a three-dimensional effect. For more information, see [Customizing the Control's Borders](#).

To hide the surrounding border, set the BorderStyle property to 0 (No Border) or set the BorderWidth property to 0.

Data Type

Color

[Print](#)

[Copy](#)

[Close](#)

The following example creates a combo box control whose surrounding border is red, is 3 pixels wide, and has a dash-dot style.

C

```
/* Set border color */
/* red RGB(255, 0, 0) */
LC_SetBorderColor(hWnd, 0x000000FF);
/* Set border width to 3 pixels */
LC_SetBorderWidth(hWnd, 3);
/* Set border style to dash dot */
LC_SetBorderStyle(hWnd, LC_BORDERSTYLE_DASH_DOT);
```

C++

```
// Set border color
// red RGB(255, 0, 0)
fpCombo1->SetBorderColor(0x000000FF);
// Set border width to 3 pixels
fpCombo1->SetBorderWidth(3);
// Set border style to dash dot
fpCombo1->SetBorderStyle(LC_BORDERSTYLE_DASH_DOT);
```

Visual Basic

```
' Set border color
' red RGB(255, 0, 0)
fpCombo1.BorderColor = &H000000FF&
' Set border width to 3 pixels
fpCombo1.BorderWidth = 3
' Set border style to dash dot
fpCombo1.BorderStyle = LC_BORDERSTYLE_DASH_DOT
```

See Also

[Customizing the Control's Borders](#)

[Specifying the Border Appearance](#)

[Appearance](#), [BorderStyle](#), [BorderWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

BorderDropShadow Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether and when a control displays a drop shadow.

Syntax

C **UINT** LC_GetBorderDropShadow(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetBorderDropShadow(**HWND** hWnd, **short** value);

C++ **short** **Class**::GetBorderDropShadow(**void**);
Class::SetBorderDropShadow(**short** value);

Visual Basic [form.]control.BorderDropShadow[= setting%]

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - None	(Default) No drop shadow displayed	LC_BORDERDROPSHADOW_NONE
1 - Always	Drop shadow always displayed	LC_BORDERDROPSHADOW_ALWAYS
2 - On Focus	Drop shadow displayed when the control has the focus	LC_BORDERDROPSHADOW_ONFOCUS

When the BorderDropShadow property is set to 2 (On Focus), a drop shadow is displayed when the control has the focus. For more information about ways to display the focus on the control, see [Displaying Focus on the Control](#).

You can change the color and width of the drop shadow with the [BorderDropShadowColor](#) and [BorderDropShadowWidth](#) properties.

If the drop shadow is only displayed when the control has the focus, you can change the color of the area where the drop shadow is displayed (called the gray area). To change the gray area color, set the [BorderGrayAreaColor](#) property.

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The BorderDropShadowWidth property setting affects the area inside the control's margins.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The Enabled property must be set to True for a control to receive the focus when the BorderDropShadow property is set to 2 (On Focus).

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates an fpList control that displays a dark gray drop shadow when it receives the focus. The width of the drop shadow is 3 pixels.

C

```
/* Display shadow when in focus */
LC_SetBorderDropShadow(hWnd, LC_BORDERDROPSHADOW_ONFOCUS);
/* dark gray RGB(128, 128, 128) */
LC_SetBorderDropShadowColor(hWnd, 0x00C0C0C0);
LC_SetBorderDropShadowWidth(hWnd, 3);
```

C++

```
// Display shadow when in focus
fpList1->SetBorderDropShadow(LC_BORDERDROPSHADOW_ONFOCUS);
// dark gray RGB(128, 128, 128)
fpList1->SetBorderDropShadowColor(0x00C0C0C0);
fpList1->SetBorderDropShadowWidth(3);
```

Visual Basic

```
' Display shadow when in focus
fpList1.BorderDropShadow = LC_BORDERDROPSHADOW_ONFOCUS
' dark gray RGB(128, 128, 128)
fpList1.BorderDropShadowColor = &H00C0C0C0&
fpList1.BorderDropShadowWidth = 3
```

See Also

[Displaying Focus on the Control](#)

[Displaying Drop Shadows](#)

[BorderDropShadowColor](#), [BorderDropShadowWidth](#), [BorderGrayAreaColor](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

BorderDropShadowColor Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the color of a control's drop shadow.

Syntax

C *UINT* LC_GetBorderDropShadowColor(*HWND* hWnd, *COLORREF FAR* *lpValue);
UINT LC_SetBorderDropShadowColor(*HWND* hWnd, *COLORREF* value);

C++ *COLORREF* *Class*::GetBorderDropShadowColor(*void*);
Class::SetBorderDropShadowColor(*COLORREF* value);

Visual Basic [form.]control.BorderDropShadowColor[= color]

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The default value for the BorderDropShadowColor property is &H80000010& (Windows system button shadow color).

To display a drop shadow, set the [BorderDropShadow](#) property to 1 (Always) or 2 (On Focus). The BorderDropShadowColor property specifies the color of the drop shadow.

If the BorderDropShadow property is set to 2 (On Focus), when the control does not have the focus, the area where the drop shadow color is displayed is the color specified by the [BorderGrayAreaColor](#) property.

Data Type

Color

[Print](#)

[Copy](#)

[Close](#)

The following example creates an fpList control that displays a custom drop shadow upon receiving the focus. Note the difference between the drop shadow area and the border gray area.

C

```
LC_SetBorderDropShadow(hWnd, LC_BORDERDROPSHADOW_ONFOCUS);
/* dark gray RGB(128, 128, 128) */
LC_SetBorderDropShadowColor(hWnd, 0x00C0C0C0);
LC_SetBorderDropShadowWidth(hWnd, 3);
/* red RGB(255, 0, 0) */
LC_SetBorderGrayAreaColor(hWnd, 0x000000FF);
```

C++

```
fpList1->SetBorderDropShadow(LC_BORDERDROPSHADOW_ONFOCUS);
// dark gray RGB(128, 128, 128)
fpList1->SetBorderDropShadowColor(0x00C0C0C0);
fpList1->SetBorderDropShadowWidth(3);
// red RGB(255, 0, 0)
fpList1->SetBorderGrayAreaColor(0x000000FF);
```

Visual Basic

```
fpList1.BorderDropShadow = LC_BORDERDROPSHADOW_ONFOCUS
' dark gray RGB(128, 128, 128)
fpList1.BorderDropShadowColor = &H00C0C0C0&
fpList1.BorderDropShadowWidth = 3
' red RGB(255, 0, 0)
fpList1.BorderGrayAreaColor = &H000000FF&
```

See Also

[Displaying Drop Shadows](#)

[BorderDropShadow](#), [BorderDropShadowWidth](#), [BorderGrayAreaColor](#), [ThreeDOnFocusInvert](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

BorderDropShadowWidth Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the width in pixels of the drop shadow.

Syntax

C **UINT** LC_GetBorderDropShadowWidth(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetBorderDropShadowWidth(**HWND** hWnd, **short** value);

C++ **short** **Class**::GetBorderDropShadowWidth(**void**);
Class::SetBorderDropShadowWidth(**short** value);

Visual Basic [form.]control.BorderDropShadowWidth[= value%]

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The default value for the BorderDropShadowWidth property is 3 pixels.

To display a drop shadow, set the [BorderDropShadow](#) property to 1 (Always) or 2 (On Focus). The [BorderDropShadowColor](#) property specifies the color of the drop shadow.

Note Before you set the properties that size the client area of the control and the control itself, be sure to consider the BorderDropShadowWidth property setting.

Data Type

Integer

See Also

[Displaying Drop Shadows](#)

[BorderDropShadow](#), [BorderDropShadowColor](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

BorderGrayAreaColor Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the border's gray area color to distinguish it from the form's [BackColor](#).

Syntax

C *UINT* LC_GetBorderGrayAreaColor(*HWND* hWnd, *COLORREF FAR* *pValue);
UINT LC_SetBorderGrayAreaColor(*HWND* hWnd, *COLORREF* value);

C++ *COLORREF* *Class*::GetBorderGrayAreaColor(*void*);
Class::SetBorderGrayAreaColor(*COLORREF* value);

Visual Basic [form.]control.BorderGrayAreaColor[= color]

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The default value for the BorderGrayAreaColor property is &H8000000B& (the default color which is light gray).

If you change the surrounding border's gray area color, you can see it in the following circumstances:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

When you create a drop shadow for the control by setting the [BorderDropShadow](#) property to 2 (On Focus), you can see the color when the control does not have the focus.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

When you create rounded corners by setting the [BorderStyle](#) property to 2 (Rounded) and setting the [BorderWidth](#) property to a value greater than 3 pixels, you can see the color in the corners of the control.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

When you create a three-dimensional frame by setting the [ThreeDFrameWidth](#), [ThreeDInsideStyle](#), and [ThreeDOutsideStyle](#) properties, changing the surrounding border's gray area color keeps the outer border's highlight color from defaulting to the background color of the form.

Data Type

Color

[Print](#)

[Copy](#)

[Close](#)

The following example creates a drop-down list combo box control with three columns. A blue drop shadow is displayed on focus. The border gray area color is red.

C

```
LC_SetStyle(hWnd, CBX_STYLE_DROPDOWN_LIST);
LC_SetColumns(hWnd, 3);
LC_SetBorderDropShadow(hWnd, LC_BORDERDROPSHADOW_ONFOCUS);
/* blue RGB(0, 0, 255) */
LC_SetBorderDropShadowColor(hWnd, 0x00FF0000);
/* red RGB(255, 0, 0) */
LC_SetBorderGrayAreaColor(hWnd, 0x000000FF);
```

C++

```
fpCombo1->SetStyle(CBX_STYLE_DROPDOWN_LIST);
fpCombo1->SetColumns(3);
fpCombo1->SetBorderDropShadow(LC_BORDERDROPSHADOW_ONFOCUS);
// blue RGB(0, 0, 255)
fpCombo1->SetBorderDropShadowColor(0x00FF0000);
// red RGB(255, 0, 0)
fpCombo1->SetBorderGrayAreaColor(0x000000FF);
```

Visual Basic

```
fpCombo1.Style = CBX_STYLE_DROPDOWN_LIST
fpCombo1.Columns = 3
fpCombo1.BorderDropShadow = LC_BORDERDROPSHADOW_ONFOCUS
' blue RGB(0, 0, 255)
fpCombo1.BorderDropShadowColor = &H00FF0000&
' red RGB(255, 0, 0)
fpCombo1.BorderGrayAreaColor = &H000000FF&
```

See Also

[Displaying Drop Shadows](#)

[BorderDropShadow](#), [BorderDropShadowColor](#), [BorderStyle](#), [BorderWidth](#), [ThreeDFrameWidth](#), [ThreeDInsideStyle](#), [ThreeDOutsideStyle](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

BorderStyle Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the style of the outline border.

Syntax

C *UINT* LC_GetBorderStyle(*HWND* hWnd, *short FAR* *lpValue);
UINT LC_SetBorderStyle(*HWND* hWnd, *short* value);

C++ *short* *Class*::GetBorderStyle(*void*);
Class::SetBorderStyle(*short* value);

Visual Basic [form.]control.BorderStyle[= setting%]

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0	No Border	LC_BORDERSTYLE_NO_BORDER
1	(Default) Single Line	LC_BORDERSTYLE_SINGLE_LINE
2	Rounded	LC_BORDERSTYLE_ROUNDED
3	Dash	LC_BORDERSTYLE_DASH
4	Dot	LC_BORDERSTYLE_DOT
5	Dash Dot	LC_BORDERSTYLE_DASH_DOT
6	Dash Dot Dot	LC_BORDERSTYLE_DASH_DOT_DOT

Every control has an outline border. In addition, controls can display inner and outer borders that create a three-dimensional effect. Settings 1 (Single Line) or 2 (Rounded) of the BorderStyle property are recommended when creating a three-dimensional effect for a control. For more information, see [Customizing the Control's Borders](#).

To hide the surrounding border, set the BorderStyle property to 0 (No Border) or set the [BorderWidth](#) property to 0. The BorderWidth property must be set to a value greater than 0 for the BorderStyle property to have an effect.

Data Type

Integer (Enumerated)

See Also

[Customizing the Control's Borders](#)

[Specifying the Border Appearance](#)

[Appearance](#), [BorderColor](#), [BorderWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

BorderWidth Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the width in pixels of the outline border.

Syntax

C *UINT* LC_GetBorderWidth(*HWND* hWnd, *short FAR* *lpValue);
UINT LC_SetBorderWidth(*HWND* hWnd, *short* value);

C++ *short Class*::GetBorderWidth(*void*);
Class::SetBorderWidth(*short* value);

Visual Basic [form.]control.BorderWidth[= value%]

Designer Page

[Border subtab of the Appearance designer page](#)

Remarks

The default value for the BorderWidth property is 1 pixel.

Every control has an outline border. In addition, controls can display inner and outer borders that create a three-dimensional effect. For more information, see [Customizing the Control's Borders](#).

Setting the BorderWidth property to 0 hides the outline border; if the control does not display an outline border, the [BorderColor](#) and [BorderStyle](#) properties have no effect.

Data Type

Integer

See Also

[Customizing the Control's Borders](#)

[Specifying the Border Appearance](#)

[Appearance](#), [BorderColor](#), [BorderStyle](#), [ThreeDInsideWidth](#), [ThreeDOutsideWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Col Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the index number of a column in an fpCombo or fpList control.

Syntax

C **UINT** LC_GetCol(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetCol(**HWND** hWnd, **short** value);

C++ **shortClass::**GetCol(**void**);
Class::SetCol(**short** value);

Visual Basic [form.]control.Col[= value%]

Designer Pages

Col drop-down list box on:

- [Specific subtab of the Columns designer page](#)
- [List subtab of the ApplyTo designer page](#)
- [Line subtab of the ApplyTo designer page](#)
- [Add Data designer page](#)
- [Column subtab of the Data Binding designer page](#)
- [Merge/Join designer page](#)
- [Sort designer page](#)
- [Search designer page](#)

Remarks

The default value for the Col property is 0.

The Col property specifies the column on which the designated-column properties (such as [AlignH](#) and [ColSorted](#)) operate. Column numbers start with 0, which designates the top, leftmost column. Columns are numbered from left to right and top to bottom within their parent group, if any, and then within the control.

Column index numbers are based on the physical position of the column in the control. For example, assume you define three columns (0, 1, and 2). These columns appear in that order from left to right across the top of the control. If you move Column 2 to the far left side of the control, this column now has a column index number of 0. For more information on columns, see [Referencing a Column](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To specify an entire column, set the [Row](#) property to 1 before setting the Col property.

You must set the [Columns](#) property to create a multiple-column fpCombo or fpList control before setting the Col property.

Note No column properties, including Col, apply when the [WrapList](#) property is set to True, regardless of the Columns property value.

You can set the Col property using the FarPoint Property Designer. For more information, see [Using the FarPoint Property Designer](#).

You can use the [ColID](#) and [ColName](#) properties to assign unique identifiers to a column. You can then use the [ColFromID](#) or [ColFromName](#) properties to specify the column to which the designated-column properties apply, regardless of where the column physically appears in the control.

Tip Because column index numbers are based on the physical position of the column in the control, if you plan on moving columns or changing column levels, or allowing the user to move columns, we strongly recommend you use one of the unique column identifiers (ColID or ColName property) to reference a column rather than the Col property.

Data Type

Integer

See Also

[Referencing a Column](#)

[Using the FarPoint Property Designer](#)

[AlignH](#), [ColDataField](#), [ColFormat](#), [ColFromID](#), [ColFromName](#), [ColHeaderText](#), [ColHide](#), [ColID](#), [ColLevel](#), [ColLevelHeight](#), [ColList](#), [ColLockResize](#), [ColMerge](#), [ColName](#), [ColParentGroup](#), [ColPos](#), [ColPosInParent](#), [ColSorted](#), [ColSortSeq](#), [ColText](#), [Columns](#), [ColWidth](#), [ExtendCol](#), [Row](#), [WrapList](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColDataField Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the data field to bind to a column in a multiple-column fpCombo or fpList control. This property is available for Visual Basic users only.

Syntax

Visual Basic `[form.]control.ColDataField[= text$]`

Designer Pages

[Column subtab of the Data Binding designer page](#)

Remarks

Use the ColDataField property with multiple-column fpCombo and fpList controls. The ColDataField property lets you bind each column to a separate field in the database bound to the Data control.

Before you set the ColDataField property, you must create a multiple-column control with the [Columns](#) property and specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.

Instead of specifying the field name as the value, you can use the field number, written as "*#field number*". For example, instead of using

```
fpCombo1.ColDataField = "Address"
```

you could use

```
fpCombo1.ColDataField = "#2"
```

The field number is zero-based.

Use the DataField property with single-column controls. Use the [DataFieldList](#) property to bind the list in a single-column fpCombo control.

For more information on how to bind a data field, refer to the DataField property in the Visual Basic documentation.

Data Type

String

See Also

[Providing Column Headers](#)

[Working with Databases](#)

[Col](#), [ColFromID](#), [ColFromName](#), [Columns](#), [DataFieldList](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColFormat Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the format string used for data display in a column. This property applies only when reading bound data. This property is available for Visual Basic users only.

Syntax

Visual Basic `[form.]control.ColFormat[= text$]`

Designer Pages

[Specific subtab of the Columns designer page](#)

[Column subtab of the Data Binding designer page](#)

Remarks

The format string specified with the ColFormat property affects the way data is stored in the data field.

Before you set the ColFormat property, you must create a multiple-column control with the [Columns](#) property and specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.

See the Visual Basic Format\$ function for valid format strings.

Data Type

String

[Print](#)

[Copy](#)

[Close](#)

The following example creates a two-column fpCombo control that formats the data in the first column. To re-create this example, create an fpCombo control and a Data control on a form. Bind the Data control to Visual Basic's BIBLIO.MDB database using the DatabaseName property and to the Titles table using the RecordSource property. Then bind the fpCombo control to the Data control using the DataSource and [DataSourceList](#) properties.

Visual Basic

```
fpCombo1.Columns = 2
' Display second column, "Title," in edit field
fpCombo1.ColumnEdit = 1
fpCombo1.DataAutoHeadings = False
fpCombo1.ColumnHeaderShow = True
fpCombo1.Col = 0
fpCombo1.ColDataField = "Year Published"
fpCombo1.ColHeaderText = "Year"
fpCombo1.ColFormat = "(####)"
fpCombo1.Col = 1
fpCombo1.ColDataField = "Title"
fpCombo1.ColHeaderText = "Title of Book"
```

See Also

[Customizing Columns](#)

[Col](#), [ColFromID](#), [ColFromName](#), [Columns](#), [DataSourceList](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColFromID Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the column on which the designated-column properties operate by using the column identifier number.

Syntax

C *UINT* LC_GetColFromID(*HWND* hWnd, *long FAR **lpValue);
 UINT LC_SetColFromID(*HWND* hWnd, *long* value);

C++ *long Class::*GetColFromID(*void*);
 *Class::*SetColFromID(*long* value);

Visual Basic [*form.*]control.ColFromID[= *value*&]

Designer Pages

Col ID drop-down list box on:

- [Specific subtab of the Columns designer page](#)
- [List subtab of the ApplyTo designer page](#)
- [Line subtab of the ApplyTo designer page](#)
- [Add Data designer page](#)
- [Column subtab of the Data Binding designer page](#)
- [Merge/Join designer page](#)
- [Sort designer page](#)
- [Search designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the ColFromID property is 1.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To specify an entire column, set the [Row](#) property to 1 before setting the ColFromID property.

You must set the [ColID](#) property to create the column identifier number before setting the ColFromID property. Once you define an identifier number for a column, you can use the ColFromID property to specify the column on which the designated-column properties (such as [ColHeaderText](#) and [ColSorted](#)) operate. The ColFromID property works the same as the [Col](#) property in this respect.

You can also use the [ColName](#) and [ColFromName](#) properties in a similar fashion.

Tip Because column index numbers are based on the physical position of the column in the control, if you plan on moving columns or changing column levels, we strongly recommend you use one of the unique column identifiers (ColID or ColName property) to reference a column rather than the Col property.

Note No column properties, including ColFromID, apply when the [WrapList](#) property is set to True, regardless of the Columns property value.

Data Type

Integer (Long)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a two-column list box with two levels in every row. The first column's height is two levels.

C

```
LC_SetColumns(hWnd, 2);
LC_SetColumnHeaderShow(hWnd, TRUE);
LC_SetColumnLevels(hWnd, 2);
/* Insert data */
LC_SetInsertRow(hWnd, "SoftTech\tActiveX controls");
LC_SetInsertRow(hWnd, "FarOut\tVBX controls");
LC_SetInsertRow(hWnd, "OldHat\tDLL control");
/* Define columns */
LC_SetCol(hWnd, 0);
LC_SetColID(hWnd, 111);
LC_SetCol(hWnd, 1);
LC_SetColID(hWnd, 112);
/* Set level height and text for first column */
LC_SetColFromID(hWnd, 111);
LC_SetColLevelHeight(hWnd, 2);
LC_SetColHeaderText(hWnd, "Company");
/* Set level height and text for second column */
LC_SetColFromID(hWnd, 112);
LC_SetColHeaderText(hWnd, "Product");
```

C++

```
fpList1->SetColumns(2);
fpList1->SetColumnHeaderShow(TRUE);
fpList1->SetColumnLevels(2);
// Insert data
fpList1->SetInsertRow("SoftTech\tActiveX controls");
fpList1->SetInsertRow("FarOut\tVBX controls");
fpList1->SetInsertRow("OldHat\tDLL control");
// Define columns
fpList1->SetCol(0);
fpList1->SetColID(111);
fpList1->SetCol(1);
fpList1->SetColID(112);
// Set level height and text for first column
fpList1->SetColFromID(111);
fpList1->SetColLevelHeight(2);
fpList1->SetColHeaderText("Company");
// Set level height and text for second column
fpList1->SetColFromID(112);
fpList1->SetColHeaderText("Product");
```

Visual Basic

```
fpList1.Columns = 2
fpList1.ColumnHeaderShow = True
fpList1.ColumnLevels = 2
' Insert data
fpList1.InsertRow = "SoftTech" & Chr$(9) & "ActiveX controls"
fpList1.InsertRow = "FarOut" & Chr$(9) & "VBX controls"
fpList1.InsertRow = "OldHat" & Chr$(9) & "DLL control"
' Define columns
fpList1.Col = 0
fpList1.ColID = 111
fpList1.Col = 1
fpList1.ColID = 112
' Set level height and text for first column
fpList1.ColFromID = 111
fpList1.ColLevelHeight = 2
fpList1.ColHeaderText = "Company"
```


' Set level height and text for second column Applying Properties to a Specific Column
fpList1.ColFromID = 112
fpList1.ColHeaderText = "Product"

See Also

[Applying Properties to a Specific Column](#)

[Referencing a Column](#)

[Col](#), [ColDataField](#), [ColFormat](#), [ColFromName](#), [ColHeaderText](#), [ColHide](#), [ColID](#), [ColLevel](#), [ColLevelHeight](#), [ColList](#), [ColLockResize](#), [ColMerge](#), [ColName](#), [ColParentGroup](#), [ColPos](#), [ColPosInParent](#), [ColSorted](#), [ColSortSeq](#), [ColText](#), [Columns](#), [ColWidth](#), [Row](#), [WrapList](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColFromName Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the column on which the designated-column properties operate by using the column name.

Syntax

C *UINT* LC_GetColFromName(*HWND* hWnd, *LPSTR* buffer, *UINT* nBufferSize);
 UINT LC_SetColFromName(*HWND* hWnd, *LPCSTR* value);

C++ *LPSTR Class::*GetColFromName(*LPSTR* buffer, *UINT* bufferSize);
 *Class::*SetColFromName(*LPCSTR* value);

Visual Basic [form.]control.ColFromName[= text\$]

Designer Pages

Col Name drop-down list box on:

- [Specific subtab of the Columns designer page](#)
- [List subtab of the ApplyTo designer page](#)
- [Line subtab of the ApplyTo designer page](#)
- [Add Data designer page](#)
- [Column subtab of the Data Binding designer page](#)
- [Merge/Join designer page](#)
- [Sort designer page](#)
- [Search designer page](#)

Remarks

You must set the [ColName](#) property to define the column name before setting the ColFromName property. Once you define a name for a column, you can use the ColFromName property to specify the column on which the designated-column properties (such as [ColHeaderText](#) and [ColSorted](#)) operate. The ColFromName property works the same as the [Col](#) property in this respect.

The ColName property is case-sensitive; however, the ColFromName property is not case-sensitive.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To specify an entire column, set the [Row](#) property to 1 before setting the ColFromName property.

You can also use the [ColID](#) and [ColFromID](#) properties in a similar fashion.

Tip Because column index numbers are based on the physical position of the column in the control, if you plan on moving columns or changing column levels, we strongly recommend you use one of the unique column identifiers (ColID or ColName property) to reference a column rather than the Col property.

Note No column properties, including ColFromName, apply when the [WrapList](#) property is set to True, regardless of the Columns property value.

Data Type

String

[Print](#)

[Copy](#)

[Close](#)

The following example creates a three-column list box control with three levels. Each column is on a different level.

C

```
LC_SetColumnHeaderShow(hWnd, FALSE);
LC_SetColumns(hWnd, 3);
LC_SetColumnLevels(hWnd, 3);
LC_SetInsertRow(hWnd, "Bob Morris\t1994\tProgrammer");
LC_SetInsertRow(hWnd, "Rick Johnson\t1992\tWriter");
LC_SetInsertRow(hWnd, "Ann Bennett\t1995\tManager");
/* Define columns */
LC_SetCol(hWnd, 0);
LC_SetColName(hWnd, "NAME");
LC_SetCol(hWnd, 1);
LC_SetColName(hWnd, "DOH");
LC_SetCol(hWnd, 2);
LC_SetColName(hWnd, "TITL");
/* Define column levels */
LC_SetColFromName(hWnd, "NAME");
LC_SetColLevel(hWnd, 0);
LC_SetColFromName(hWnd, "TITL");
LC_SetColLevel(hWnd, 1);
LC_SetColFromName(hWnd, "DOH");
LC_SetColLevel(hWnd, 2);
```

C++

```
fpList1->SetColumnHeaderShow(FALSE);
fpList1->SetColumns(3);
fpList1->SetColumnLevels(3);
fpList1->SetInsertRow("Bob Morris\t1994\tProgrammer");
fpList1->SetInsertRow("Rick Johnson\t1992\tWriter");
fpList1->SetInsertRow("Ann Bennett\t1995\tManager");
// Define columns
fpList1->SetCol(0);
fpList1->SetColName("NAME");
fpList1->SetCol(1);
fpList1->SetColName("DOH");
fpList1->SetCol(2);
fpList1->SetColName("TITL");
// Define column levels
fpList1->SetColFromName("NAME");
fpList1->SetColLevel(0);
fpList1->SetColFromName("TITL");
fpList1->SetColLevel(1);
fpList1->SetColFromName("DOH");
fpList1->SetColLevel(2);
```

Visual Basic

```
fpList1.ColumnHeaderShow = False
fpList1.Columns = 3
fpList1.ColumnLevels = 3
fpList1.InsertRow = "Bob Morris" & Chr$(9) & 1994 & Chr$(9) & "Programmer"
fpList1.InsertRow = "Rick Johnson" & Chr$(9) & 1992 & Chr$(9) & "System Analyst"
fpList1.InsertRow = "Ann Bennett" & Chr$(9) & 1995 & Chr$(9) & "Manager"
' Define columns
fpList1.Col = 0
fpList1.ColName = "NAME"
fpList1.Col = 1
fpList1.ColName = "DOH"
fpList1.Col = 2
fpList1.ColName = "TITL"
' Define column levels
fpList1.ColFromName = "NAME"
```

```
fpList1.ColLevel = 0
fpList1.ColFromName = "TITL"
fpList1.ColLevel = 1
fpList1.ColFromName = "DOH"
fpList1.ColLevel = 2
```

See Also

[Applying Properties to a Specific Column](#)

[Referencing a Column](#)

[Col](#), [ColDataField](#), [ColFormat](#), [ColFromID](#), [ColHeaderText](#), [ColHide](#), [ColID](#), [ColLevel](#), [ColLevelHeight](#), [ColList](#), [ColLockResize](#), [ColMerge](#), [ColName](#), [ColParentGroup](#), [ColPos](#), [ColPosInParent](#), [ColSorted](#), [ColSortSeq](#), [ColText](#), [Columns](#), [ColWidth](#), [Row](#), [WrapList](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColHeaderText Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns a string to display in the list header of an fpCombo or fpList control.

Syntax

C	UINT LC_GetColHeaderText(<i>HWND</i> hWnd, <i>LPSTR</i> buffer, <i>UINT</i> nBufferSize); UINT LC_SetColHeaderText(<i>HWND</i> hWnd, <i>LPCSTR</i> value);
C++	LPSTR Class:: GetColHeaderText(<i>LPSTR</i> buffer, <i>UINT</i> bufferSize); Class:: SetColHeaderText(<i>LPCSTR</i> value);
Visual Basic	[form.]control.ColHeaderText[= text\$]

Designer Page

[Specific subtab of the Columns designer page](#)

Remarks

The default value for the ColHeaderText property is an empty string.

You must set the [ColumnHeaderShow](#) property to True to display the headers.

To display a user-defined header in a single-column, non-bound fpCombo or fpList control, set the [Columns](#) property to 1, the [Col](#) property to 0, and the ColumnHeaderShow property to True before setting the ColHeaderText property. If you are working with a single-column, bound control, do not set the DataField property for the single column. Specify the field to bind to the column using the [ColDataField](#) property.

By default, when each column of a multiple-column fpCombo or fpList control is bound to a database, the field name is displayed as the header text. To display other header text, you must set the [DataAutoHeadings](#) property to False and the ColumnHeaderShow property to True.

To display a user-defined header in multiple-column, non-bound fpCombo or fpList controls, create a multiple-column control with the Columns property, specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property, and define the header text with the ColHeaderText property.

Data Type

String

See Also

[Providing Column Headers](#)

[Col](#), [ColDataField](#), [ColFromID](#), [ColFromName](#), [ColumnHeaderShow](#), [Columns](#), [DataAutoHeadings](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColHide Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether to hide a column in a multiple-column fpCombo or fpList control. This property is available at run time only.

Syntax

C **UINT** LC_GetColHide(**HWND** hWnd, **BOOL FAR** *pValue);
UINT LC_SetColHide(**HWND** hWnd, **BOOL** value);

C++ **BOOL Class::**GetColHide(**void**);
Class::SetColHide(**BOOL** value);

Visual Basic [form.]control.ColHide[= boolean%]

Designer Page

[Specific subtab of the Columns designer page](#)

Remarks

The default value for the ColHide property is False.

Use this property when you want to sort the values in a column without displaying the column by which values are sorted. You might also want to use this property to provide the values to display in the edit field, as set by the [ColumnEdit](#) property, when you do not want those values displayed in a list column.

Before you set the ColHide property, you must specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.

Tip Another way to hide a column is to set the [ColWidth](#) property to 0.

Data Type

Integer (Boolean)

See Also

[Customizing Columns](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColumnEdit](#), [ColWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColID Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the column identifier number.

Syntax

C *UINT* LC_GetColID(*HWND* hWnd, *long FAR *lpValue*);
 UINT LC_SetColID(*HWND* hWnd, *long value*);

C++ *long Class::GetColID(void)*;
 Class::SetColID(long value);

Visual Basic [form.]control.ColID[= value&]

Designer Page

[Specific subtab of the Columns designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the ColID property is 1.

You must set the [Col](#) property to specify the column before setting the ColID property.

The ColID property defines a unique identifier number for a column. Once you define an identifier number for a column, you can use the [ColFromID](#) property to specify the column on which the designated-column properties (such as [ColHeaderText](#) and [ColSorted](#)) operate. The ColFromID property works the same as the Col property in this respect.

You can also use the [ColName](#) and [ColFromName](#) properties in a similar fashion.

Tip Because column index numbers are based on the physical position of the column in the control, if you plan on moving columns or changing column levels, we strongly recommend you use one of the unique column identifiers (ColID or ColName property) to reference a column rather than the Col property.

Note No column properties, including ColID, apply when the [WrapList](#) property is set to True, regardless of the Columns property value.

Data Type

Integer (Long)

See Also

[Applying Properties to a Specific Column](#)

[Referencing a Column](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColHeaderText](#), [ColName](#), [ColSorted](#), [WrapList](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColLevel Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the column level within a row.

Syntax

C	UINT LC_GetColLevel(HWND hWnd, short FAR *lpValue); UINT LC_SetColLevel(HWND hWnd, short value);
C++	short Class:: GetColLevel(void); Class:: SetColLevel(short value);
Visual Basic	[form.]control.ColLevel[= value%]

Designer Page

[Specific subtab of the Columns designer page](#)

Remarks

Within a row you can have multiple levels of columns. The column levels are zero-based. The first level within a row is level 0.

The default value for the ColLevel property is 0, which displays all columns in a row on the same line.

Use the ColLevel property to define the level within a row where a column is displayed. Columns continue to be displayed on the same level until you set the ColLevel property. For example, assume that an fpList control has three column levels and three columns. Also assume that you want to display one column on each level within a row. To accomplish this, first define the columns. Then for each column, set the ColLevel property to 0, 1, and 2, respectively (see [Example](#)).

Use the [ColumnLevels](#) property to specify the number of levels within each row of the control. Use the [ColLevelHeight](#) property to define the height of a column in levels.

Before you set the ColLevel property, you must specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.

Tips Define all columns (Col, [ColID](#), and [ColName](#) properties) in the control before moving columns with the ColLevel property.

Because column index numbers are based on the physical position of the column in the control, if you plan on moving columns or changing column levels, we strongly recommend you use one of the unique column identifiers (ColID or ColName property) to reference a column rather than the Col property.

Data Type

Integer

See Also

[Creating Levels of Columns Within a Row](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColID](#), [ColLevelHeight](#), [ColName](#), [ColumnLevels](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColLevelHeight Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the height of a column in terms of levels.

Syntax

C	UINT LC_GetColLevelHeight(HWND hWnd, short FAR *lpValue); UINT LC_SetColLevelHeight(HWND hWnd, short value);
C++	short Class ::GetColLevelHeight(void); Class ::SetColLevelHeight(short value);
Visual Basic	[form.]control.ColLevelHeight[= value%]

Designer Page

[Specific subtab of the Columns designer page](#)

Remarks

The default value for the ColLevelHeight property is 1 level. The height of a level is equal to the row height.

The column level height applies to the same column in every row. For example, if you set the column level height of the first column to 2 levels, the first column of every row will have a column level height of 2 levels.

Before you set the ColLevelHeight property, you must specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.

Use the [ColumnLevels](#) property to specify the number of levels within each row in the control. The column level height cannot exceed this number.

Data Type

Integer

See Also

[Creating Levels of Columns Within a Row](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColLevel](#), [ColumnLevels](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColList Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns a value in the specified row and column from the list in an fpCombo or fpList control. This property is available at run time only.

Syntax

C **UINT** LC_GetColList(**HWND** hWnd, **LPSTR** buffer, **UINT** nBufferSize);
UINT LC_SetColList(**HWND** hWnd, **LPCSTR** value);

C++ **LPSTR Class::**GetColList(**LPSTR** buffer, **UINT** bufferSize);
Class::SetColList(**LPCSTR** value);

Visual Basic [form.]control.ColList[= text\$]

Designer Page

[Add Data designer page](#)

Remarks

Use the ColList property to set or return a list item from a multiple-column fpCombo or fpList control. You must specify the column and row with either the [Col](#), [ColFromID](#), or [ColFromName](#) property and the [Row](#) property before you set the ColList property.

You can use the ColList property to add data to a nonbound column in a multiple-column bound fpCombo or fpList control. For more information, see [Adding Data to Nonbound Columns](#).

The default value for the ColList property is an empty string.

Note Use the [List](#) property to return a list item from a single-column fpCombo or fpList control.

Data Type

String

[Print](#)

[Copy](#)

[Close](#)

The following example creates a two-column fpCombo control. List items are added to the second column during the Form_Load event. The first column is displayed in the edit field for user entry.

C

```
LC_SetColumns(hWnd, 2);
Cbx_SetColumnEdit(hWnd, 0);
LC_SetColumnHeaderShow(hWnd, TRUE);
LC_SetCol(hWnd, 0);
LC_SetColHeaderText(hWnd, "Inventory Date");
LC_SetColWidth(hWnd, 25);
LC_SetCol(hWnd, 1);
LC_SetColHeaderText(hWnd, "Title");
LC_SetCol(hWnd, 1);
LC_SetRow(hWnd, 0);
LC_SetColList(hWnd, "Ben Hur");
LC_SetCol(hWnd, 1);
LC_SetRow(hWnd, 1);
LC_SetColList(hWnd, "The Great Gatsby");
```

C++

```
fpCombo1->SetColumns(2);
fpCombo1->SetColumnEdit(0);
fpCombo1->SetColumnHeaderShow(TRUE);
fpCombo1->SetCol(0);
fpCombo1->SetColHeaderText("Inventory Date");
fpCombo1->SetColWidth(25);
fpCombo1->SetCol(1);
fpCombo1->SetColHeaderText("Title");
fpCombo1->SetCol(1);
fpCombo1->SetRow(0);
fpCombo1->SetColList("Ben Hur");
fpCombo1->SetCol(1);
fpCombo1->SetRow(1);
fpCombo1->SetColList("The Great Gatsby");
```

Visual Basic

```
fpCombo1.Columns = 2
fpCombo1.ColumnEdit = 0
fpCombo1.ColumnHeaderShow = True
fpCombo1.Col = 0
fpCombo1.ColHeaderText = "Inventory Date"
fpCombo1.ColWidth = 25
fpCombo1.Col = 1
fpCombo1.ColHeaderText = "Title"
fpCombo1.Col = 1
fpCombo1.Row = 0
fpCombo1.ColList = "Ben Hur"
fpCombo1.Col = 1
fpCombo1.Row = 1
fpCombo1.ColList = "The Great Gatsby"
```

See Also

[Accessing List Items](#)

[Adding Data to Nonbound Columns](#)

[Col](#), [ColFromID](#), [ColFromName](#), [List](#), [Row](#) properties

[DataLoaded](#) event

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColLockResize Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether a column in a multiple-column fpCombo or fpList control is locked and cannot be resized by dragging the column border.

Syntax

C **UINT** LC_GetColLockResize(**HWND** hWnd, **BOOL FAR** *pValue);
UINT LC_SetColLockResize(**HWND** hWnd, **BOOL** value);

C++ **BOOL** **Class**::GetColLockResize(**void**);
Class::SetColLockResize(**BOOL** value);

Visual Basic [form.]control.ColLockResize[= *boolean%*]

Designer Page

[Specific subtab of the Columns designer page](#)

Remarks

When the [AllowColResize](#) property is set to a value greater than zero, the user can resize columns by dragging their borders. When set to True, the ColLockResize property locks a specific column, preventing it from being resized by dragging the column border.

The default value for the ColLockResize property is False. Set the ColLockResize property to True to prevent the user from dragging the right border of a column to resize it.

Before you set the ColLockResize property, you must specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.

Data Type

Integer (Boolean)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a six-column fpList control. The first two columns are frozen and the third column is locked against resizing.

C

```
LC_SetColumns(hWnd, 6);
LC_SetAllowColResize(hWnd, LC_ALLOWCOLRESIZE_RESIZECOLORHEADER);
/* Lock column 3 */
LC_SetCol(hWnd, 2);
LC_SetColLockResize(hWnd, TRUE);
/* Freeze the first two columns from scrolling */
LC_SetColsFrozen(hWnd, 2);
```

C++

```
fpList1->SetColumns(6);
fpList1->SetAllowColResize(LC_ALLOWCOLRESIZE_RESIZECOLORHEADER);
// Lock column 3
fpList1->SetCol(2);
fpList1->SetColLockResize(TRUE);
// Freeze the first two columns from scrolling
fpList1->SetColsFrozen(2);
```

Visual Basic

```
fpList1.Columns = 6
fpList1.AllowColResize = LC_ALLOWCOLRESIZE_RESIZECOLORHEADER
' Lock column 3
fpList1.Col = 2
fpList1.ColLockResize = True
' Freeze the first two columns from scrolling
fpList1.ColsFrozen = 2
```

See Also

[Customizing Columns](#)

[Resizing Columns](#)

[AllowColResize](#), [Col](#), [ColFromID](#), [ColFromName](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColMerge Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether and how cells within a column with the same contents should be grouped in a single cell spanning multiple rows.

Syntax

C *UINT* LC_GetColMerge(*HWND* hWnd, *short FAR* *lpValue);
UINT LC_SetColMerge(*HWND* hWnd, *short* value);

C++ *short Class::*GetColMerge(*void*);
*Class::*SetColMerge(*short* value);

Visual Basic [form.]control.ColMerge[= setting%]

Designer Page

[Merge/Join designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Off	(Default) Cells that contain the same text are not merged	LC_COLMERGE_OFF
1 - Always	Cells that contain the same text are always merged	LC_COLMERGE_ALWAYS
2 - Restricted	Cells that contain the same text are merged only when adjacent cells to the left are also merged	LC_COLMERGE_RESTRICTED

Before you set the ColMerge property, you must specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.

Note You cannot merge columns if rows have multiple levels. If the [ColumnLevels](#) property is set to a value greater than zero, the ColMerge property is automatically set to 0 (Off).

You might use this property to reconfigure database information. For example, assume your database contains manufacturing data for two products and the normal way to display that data is by day of the week (column 0), product name (column 1), and output (column 2). You could merge column 2 and display the data by product name.

Day	Product	Line	Amount
Tuesday	Peppers	C17	7.432
	Rice	A23	2.302
	Rice	C17	6.789
Monday	Corn	D14	5.432
	Peas	C17	4.321
	Beans	C17	4.321
	Beans	A23	3.761

You could also merge column 2 and display the data by product name.

Day	Product	Line	Amount
Tuesday	Peppers	C17	7.432
	Rice	A23	2.302
Monday		C17	6.789
	Corn	D14	5.432
	Peas	C17	4.321
	Beans	C17	4.321
	A23	3.761	

Use the [RowMerge](#) property to merge cells in rows that contain the same text.

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a four-column list box control. Cells with the same text in the first and third columns are always merged. Cells with the same text in the second column are merged restrictively. The view of cell contents of the merged columns is adjusted as the user scrolls through the list.

C

```
LC_SetColumns(hWnd, 4);
LC_SetLineStyle(hWnd, LC_LINESTYLE_LOWERED);
LC_SetColumnHeaderShow(hWnd, TRUE);
LC_SetMergeAdjustView(hWnd, TRUE);
LC_SetInsertRow(hWnd, "Monday\tBeans\tA23\t3.76");
LC_SetInsertRow(hWnd, "Monday\tBeans\tC17\t4.32");
LC_SetInsertRow(hWnd, "Monday\tCorn\tC17\t4.32");
LC_SetInsertRow(hWnd, "Monday\tRice\tD14\t5.43");
LC_SetInsertRow(hWnd, "Tuesday\tRice\tC17\t6.78");
LC_SetInsertRow(hWnd, "Tuesday\tCorn\tA23\t7.41");
/* Define columns */
LC_SetCol(hWnd, 0);
LC_SetColName(hWnd, "Day");
LC_SetColHeaderText(hWnd, "Day");
LC_SetCol(hWnd, 1);
LC_SetColName(hWnd, "Prod");
LC_SetColHeaderText(hWnd, "Product");
LC_SetCol(hWnd, 2);
LC_SetColName(hWnd, "Line");
LC_SetColHeaderText(hWnd, "Line");
LC_SetCol(hWnd, 3);
LC_SetColName(hWnd, "Amt");
LC_SetColHeaderText(hWnd, "Amount");
/* Merge the first three columns */
LC_SetColFromName(hWnd, "Day");
LC_SetColMerge(hWnd, LC_COLMERGE_ALWAYS);
LC_SetColFromName(hWnd, "Prod");
LC_SetColMerge(hWnd, LC_COLMERGE_RESTRICTED);
LC_SetColFromName(hWnd, "Line");
LC_SetColMerge(hWnd, LC_COLMERGE_ALWAYS);
```

C++

```
fpList1->SetColumns(4);
fpList1->SetLineStyle(LC_LINESTYLE_LOWERED);
fpList1->SetColumnHeaderShow(TRUE);
fpList1->SetMergeAdjustView(TRUE);
fpList1->SetInsertRow("Monday\tBeans\tA23\t3.76");
fpList1->SetInsertRow("Monday\tBeans\tC17\t4.32");
fpList1->SetInsertRow("Monday\tCorn\tC17\t4.32");
fpList1->SetInsertRow("Monday\tRice\tD14\t5.43");
fpList1->SetInsertRow("Tuesday\tRice\tC17\t6.78");
fpList1->SetInsertRow("Tuesday\tCorn\tA23\t7.41");
// Define columns
fpList1->SetCol(0);
fpList1->SetColName("Day");
fpList1->SetColHeaderText("Day");
fpList1->SetCol(1);
fpList1->SetColName("Prod");
fpList1->SetColHeaderText("Product");
fpList1->SetCol(2);
fpList1->SetColName("Line");
fpList1->SetColHeaderText("Line");
fpList1->SetCol(3);
fpList1->SetColName("Amt");
fpList1->SetColHeaderText("Amount");
// Merge the first three columns
fpList1->SetColFromName("Day");
```

```

fpList1->SetColMerge(LC_COLMERGE_ALWAYS);
fpList1->SetColFromName("Prod");
fpList1->SetColMerge(LC_COLMERGE_RESTRICTED);
fpList1->SetColFromName("Line");
fpList1->SetColMerge(LC_COLMERGE_ALWAYS);

```

Visual Basic

```

fpList1.Columns = 4
fpList1.LineStyle = LC_LINESTYLE_LOWERED
fpList1.ColumnHeaderShow = True
fpList1.MergeAdjustView = True
fpList1.InsertRow = "Monday" & Chr$(9) & "Beans" & Chr$(9) & "A23" & Chr$(9) & "3.76"
fpList1.InsertRow = "Monday" & Chr$(9) & "Beans" & Chr$(9) & "C17" & Chr$(9) & "4.32"
fpList1.InsertRow = "Monday" & Chr$(9) & "Corn" & Chr$(9) & "C17" & Chr$(9) & "4.32"
fpList1.InsertRow = "Monday" & Chr$(9) & "Rice" & Chr$(9) & "D14" & Chr$(9) & "5.43"
fpList1.InsertRow = "Tuesday" & Chr$(9) & "Rice" & Chr$(9) & "C17" & Chr$(9) & "6.78"
fpList1.InsertRow = "Tuesday" & Chr$(9) & "Corn" & Chr$(9) & "A23" & Chr$(9) & "7.41"
' Define columns
fpList1.Col = 0
fpList1.ColName = "Day"
fpList1.ColHeaderText = "Day"
fpList1.Col = 1
fpList1.ColName = "Prod"
fpList1.ColHeaderText = "Product"
fpList1.Col = 2
fpList1.ColName = "Line"
fpList1.ColHeaderText = "Line"
fpList1.Col = 3
fpList1.ColName = "Amt"
fpList1.ColHeaderText = "Amount"
' Merge the first three columns
fpList1.ColFromName = "Day"
fpList1.ColMerge = LC_COLMERGE_ALWAYS
fpList1.ColFromName = "Prod"
fpList1.ColMerge = LC_COLMERGE_RESTRICTED
fpList1.ColFromName = "Line"
fpList1.ColMerge = LC_COLMERGE_ALWAYS

```

See Also

[Merging Columns or Rows](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColumnLevels](#), [MergeAdjustView](#), [RowMerge](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColName Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the column name.

Syntax

C	UINT LC_GetColName(HWND hWnd, LPSTR buffer, UINT nBufferSize); UINT LC_SetColName(HWND hWnd, LPCSTR value);
C++	LPSTR Class:: GetColName(LPSTR buffer, UINT bufferSize); Class:: SetColName(LPCSTR value);
Visual Basic	[form.]control.ColName[= text\$]

Designer Page

[Specific subtab of the Columns designer page](#)

Remarks

You must set the [Col](#) property to create the column before setting the ColName property.

The ColName property defines a unique name for a column. Once you define a name for a column, you can use the [ColFromName](#) property to specify the column on which the designated-column properties (such as [ColHeaderText](#) and [ColSorted](#)) operate. The ColFromName property works the same as the Col property in this respect.

The ColName property is case-sensitive; however, the ColFromName property is not case-sensitive.

You can also use the [ColID](#) and [ColFromID](#) properties in a similar fashion.

Tip Because column index numbers are based on the physical position of the column in the control, if you plan on moving columns or changing column levels, we strongly recommend you use one of the unique column identifiers (ColID or ColName property) to reference a column rather than the Col property.

Note No column properties, including ColName, apply when the [WrapList](#) property is set to True, regardless of the Columns property value.

Data Type

String

See Also

[Applying Properties to a Specific Column](#)

[Referencing a Column](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColHeaderText](#), [ColID](#), [ColSorted](#), [WrapList](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColParentGroup Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the parent group of a column.

Syntax

C *UINT* LC_GetColParentGroup(*HWND* hWnd, *short FAR* *lpValue);
UINT LC_SetColParentGroup(*HWND* hWnd, *short* value);

C++ *short Class::*GetColParentGroup(*void*);
*Class::*SetColParentGroup(*short* value);

Visual Basic [form.]control.ColParentGroup[= value%]

Designer Page

[Specific subtab of the Columns designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the ColParentGroup property is 1, which specifies that a column has no parent.

Before you set the ColParent property, you must specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property. Groups can have children. A child of a group can be another group ([GrpParentGroup](#) property) or a column, but not both at the same time. A column can be a child of a group that is a child of another group. For more information about groups and how they work, see [Working with Groups](#).

Children of groups exhibit the following characteristics:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If you move a group, the group's children move with it.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

When you hide a group ([GrpHide](#) property), the group's children are also hidden.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Children are automatically sized to fit the group width ([GrpWidth](#) property). This can result in text not being fully displayed in a column.

Tips

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Define all columns (Col, [ColID](#), and [ColName](#) properties) in the control before moving columns with the ColParentGroup property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Because column index numbers are based on the physical position of the column in the control, if you plan on moving columns or changing column levels, we strongly recommend you use one of the unique column identifiers (ColID or ColName property) to reference a column rather than the Col property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If you are using groups, all columns should be assigned to a group.

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example creates a three-column list box control that has two groups. The first two columns are children of group 1 and the third column is a child of group 2. The second column is in the first position in group 1 and the first column is in the second position in group 1.

C

```
LC_SetColumns(hWnd, 3);
LC_SetColumnHeaderShow(hWnd, TRUE);
LC_SetLineStyle(hWnd, LC_LINESTYLE_LOWERED);
LC_SetGroups(hWnd, 2);
LC_SetGroupHeaderShow(hWnd, TRUE);
LC_SetGrp(hWnd, 0);
LC_SetGrpHeaderText(hWnd, "Management");
LC_SetGrpWidth(hWnd, 30);
LC_SetGrp(hWnd, 1);
LC_SetGrpHeaderText(hWnd, "Human Resources");
LC_SetInsertRow(hWnd, "Bob Morris\tProgrammer\t75000");
LC_SetInsertRow(hWnd, "Rick Johnson\tSystem Analyst\t65000");
LC_SetInsertRow(hWnd, "Ann Bennett\tManager\t80000");
/* Define columns */
LC_SetCol(hWnd, 0);
LC_SetColName(hWnd, "A1");
LC_SetCol(hWnd, 1);
LC_SetColName(hWnd, "A2");
LC_SetCol(hWnd, 2);
LC_SetColName(hWnd, "A3");
/* Define position in control */
LC_SetColFromName(hWnd, "A3");
LC_SetColHeaderText(hWnd, "Salary");
LC_SetColPos(hWnd, 0);
LC_SetColFromName(hWnd, "A2");
LC_SetColHeaderText(hWnd, "Title");
LC_SetColPos(hWnd, 1);
LC_SetColFromName(hWnd, "A1");
LC_SetColHeaderText(hWnd, "Name");
LC_SetColPos(hWnd, 3);
/* Define parent group and position in parent */
/* Title column */
LC_SetColFromName(hWnd, "A2");
LC_SetColParentGroup(hWnd, 0);
LC_SetColPosInParent(hWnd, 0);
/* Name column */
LC_SetColFromName(hWnd, "A1");
LC_SetColParentGroup(hWnd, 0);
LC_SetColPosInParent(hWnd, 1);
/* Salary column */
LC_SetColFromName(hWnd, "A3");
LC_SetColParentGroup(hWnd, 1);
LC_SetColPosInParent(hWnd, 0);
LC_SetColFromName("A2");
LC_SetColWidth(15);
LC_SetColFromName("A1");
LC_SetColWidth(15);
LC_SetColFromName("A3");
LC_SetColWidth(15);
LC_SetAlignH(LC_ALIGNH_CENTER);
```

C++

```
fpList1->SetColumns(3);
fpList1->SetColumnHeaderShow(TRUE);
fpList1->SetLineStyle(LC_LINESTYLE_LOWERED);
fpList1->SetGroups(2);
fpList1->SetGroupHeaderShow(TRUE);
```



```

fpList1->SetGrp(0);
fpList1->SetGrpHeaderText("Management");
fpList1->SetGrpWidth(30);
fpList1->SetGrp(1);
fpList1->SetGrpHeaderText("Human Resources");
fpList1->SetInsertRow("Bob Morris\tProgrammer\t75000");
fpList1->SetInsertRow("Rick Johnson\tSystem Analyst\t65000");
fpList1->SetInsertRow("Ann Bennett\tManager\t80000");
// Define columns
fpList1->SetCol(0);
fpList1->SetColName("A1");
fpList1->SetCol(1);
fpList1->SetColName("A2");
fpList1->SetCol(2);
fpList1->SetColName("A3");
// Define position in control
fpList1->SetColFromName("A3");
fpList1->SetColHeaderText("Salary");
fpList1->SetColPos(0);
fpList1->SetColFromName("A2");
fpList1->SetColHeaderText("Title");
fpList1->SetColPos(1);
fpList1->SetColFromName("A1");
fpList1->SetColHeaderText("Name");
fpList1->SetColPos(3);
// Define parent group and position in parent
// Title column
fpList1->SetColFromName("A2");
fpList1->SetColParentGroup(0);
fpList1->SetColPosInParent(0);
// Name column
fpList1->SetColFromName("A1");
fpList1->SetColParentGroup(0);
fpList1->SetColPosInParent(1);
// Salary column
fpList1->SetColFromName("A3");
fpList1->SetColParentGroup(1);
fpList1->SetColPosInParent(0);
fpList1->SetColFromName("A2");
fpList1->SetColWidth(15);
fpList1->SetColFromName("A1");
fpList1->SetColWidth(15);
fpList1->SetColFromName("A3");
fpList1->SetColWidth(15);
fpList1->SetAlignH(LC_ALIGNH_CENTER);

```

Visual Basic

```

fpList1.Columns = 3
fpList1.ColumnHeaderShow = True
fpList1.LineStyle = LC_LINestyle_LOWERED
fpList1.Groups = 2
fpList1.GroupHeaderShow = True
fpList1.Grp = 0
fpList1.GrpHeaderText = "Management"
fpList1.GrpWidth = 30
fpList1.Grp = 1
fpList1.GrpHeaderText = "Human Resources"
fpList1.InsertRow = "Bob Morris" & Chr$(9) & "Programmer" & Chr$(9) & 75000
fpList1.InsertRow = "Rick Johnson" & Chr$(9) & "System Analyst" & Chr$(9) & 65000
fpList1.InsertRow = "Ann Bennett" & Chr$(9) & "Manager" & Chr$(9) & 80000
' Define columns
fpList1.Col = 0
fpList1.ColName = "A1"
fpList1.Col = 1

```

```
fpList1.ColName = "A2"  
fpList1.Col = 2  
fpList1.ColName = "A3"  
' Define position in control  
fpList1.ColFromName = "A3"  
fpList1.ColHeaderText = "Salary"  
fpList1.ColPos = 0  
fpList1.ColFromName = "A2"  
fpList1.ColHeaderText = "Title"  
fpList1.ColPos = 1  
fpList1.ColFromName = "A1"  
fpList1.ColHeaderText = "Name"  
fpList1.ColPos = 3  
' Define parent group and position in parent  
' Title column  
fpList1.ColFromName = "A2"  
fpList1.ColParentGroup = 0  
fpList1.ColPosInParent = 0  
' Name column  
fpList1.ColFromName = "A1"  
fpList1.ColParentGroup = 0  
fpList1.ColPosInParent = 1  
' Salary column  
fpList1.ColFromName = "A3"  
fpList1.ColParentGroup= 1  
fpList1.ColPosInParent = 0  
fpList1.ColFromName = "A2"  
fpList1.ColWidth = 15  
fpList1.ColFromName = "A1"  
fpList1.ColWidth = 15  
fpList1.ColFromName = "A3"  
fpList1.ColWidth = 15  
fpList1.AlignH = LC_ALIGNH_CENTER
```

See Also

[Making a Column a Child of a Group](#)

[Creating Children of Groups](#)

[Working with Groups](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColID](#), [ColName](#), [Groups](#), [GrpHide](#), [GrpLockResize](#), [GrpParentGroup](#), [GrpWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColPos Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the position number of a column within an fpCombo or fpList control.

Syntax

C *UINT* LC_GetColPos(*HWND* hWnd, *short FAR* *pValue);
UINT LC_SetColPos(*HWND* hWnd, *short* value);

C++ *short* *Class*::GetColPos(*void*);
Class::SetColPos(*short* value);

Visual Basic [form.]control.ColPos[= value%]

Designer Page

[Specific subtab of the Columns designer page](#)

Remarks

The default value for the ColPos property is 0.

Before you set the ColPos property, you must specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.

Use the [ColPosInParent](#) property to define the position number of a column within its parent. If columns are grouped, the ColPosInParent property defines the position number within the column's parent group. If columns are not grouped, the value of the ColPosInParent property is the same as the value of the ColPos property.

Tips

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Define all columns (Col, [ColID](#), and [ColName](#) properties) in the control before moving columns with the ColPos or CoPosInParent property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Because column index numbers are based on the physical position of the column in the control, if you plan on moving columns or changing column levels, we strongly recommend you use one of the unique column identifiers (ColID or ColName property) to reference a column rather than the Col property.

Note If you move a column, column index numbers will change. For more information, see [Referencing a Column](#).

Data Type

Integer

See Also

[Defining the Position of a Column Within the Control](#)

[Referencing a Column](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColID](#), [ColName](#), [ColPosInParent](#), [ColParentGroup](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColPosInParent Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the position number of a column within its parent.

Syntax

C *UINT* LC_GetColPosInParent(*HWND* hWnd, *short FAR *lpValue*);
UINT LC_SetColPosInParent(*HWND* hWnd, *short value*);

C++ *short Class::GetColPosInParent(void)*;
Class::SetColPosInParent(short value);

Visual Basic [form.]control.ColPosInParent[= value%]

Designer Page

[Specific subtab of the Columns designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the ColPosInParent property is 1 when the [Columns](#) property is set to 0.

Before you set the ColPosInParent property, you must specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property. If columns are grouped, the ColPosInParent property defines the position number within the column's parent group. Use the [ColPos](#) property to define the position number of a column within the control. If columns are not grouped, the value of the ColPosInParent property is the same as the value of the ColPos property.

Tips

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Define all columns (Col, [ColID](#), and [ColName](#) properties) in the control before moving columns with the ColPos or CoPosInParent property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Because column index numbers are based on the physical position of the column in the control, if you plan on moving columns or changing column levels, we strongly recommend you use one of the unique column identifiers (ColID or ColName property) to reference a column rather than the Col property.

Note If you move a column, column index numbers will change. For more information, see [Referencing a Column](#).

Data Type

Integer

See Also

[Defining the Position of a Column Within the Control](#)

[Referencing a Column](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColID](#), [ColName](#), [ColParentGroup](#), [ColPos](#), [Columns](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColsFrozen Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the number of columns on the left that do not scroll horizontally in a multiple-column fpCombo or fpList control.

Syntax

C *UINT* LC_GetColsFrozen(*HWND* hWnd, *short FAR* *pValue);
 UINT LC_SetColsFrozen(*HWND* hWnd, *short* value);

C++ *short Class::*GetColsFrozen(*void*);
 *Class::*SetColsFrozen(*short* value);

Visual Basic [form.]control.ColsFrozen[= value%]

Designer Page

[General subtab of the Columns designer page](#)

Remarks

The default value for the ColsFrozen property is 0.

Setting the ColsFrozen property to 1 freezes the first column (column 0, the top, leftmost column) so that it remains visible when the user clicks the horizontal scroll bar. Setting the ColsFrozen property to values greater than 1 freezes the corresponding columns. For example, setting the ColsFrozen property to 2 freezes the first and second columns.

Note If a frozen column is a child of a group and the group is not frozen, the column will not scroll horizontally. In this case, the group setting takes precedence over the column setting.

Before setting the ColsFrozen property, you must create a multiple-column fpCombo or fpList control by setting the [Columns](#) property.

Data Type

Integer

See Also

[Customizing Columns](#)

[Columns](#) property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColSortDataType Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the data type of a column in a multiple-column fpCombo or fpList control to improve sorting.

Syntax

C **UINT** LC_GetColSortDataType(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetColSortDataType(**HWND** hWnd, **short** value);

C++ **short Class::**GetColSortDataType(**void**);
Class::SetColSortDataType(**short** value);

Visual Basic [form.]control.ColSortDataType[= setting%]

Designer Page

[Sort designer page](#)

Remarks

The ColSortDataType property improves sorting in certain situations, such as when sorting by numeric values.

The following settings are available:

Setting	Description	Constant
0 - Text (No Case)	(Default) Ignores case when sorting list items	LC_COLSORTDATATYPE_TEXTNOCASE
1 - Text (Case)	Sorts list items using case (uppercase items come before lowercase items, as listed in the ANSI character set)	LC_COLSORTDATATYPE_TEXTCASE
2 - Integer	Assumes data to be integer or long and sorts in numerical order	LC_COLSORTDATATYPE_INTEGER
3 - Float	Assumes data to be floating-point numbers and sorts in numerical order, evaluating decimal values	LC_COLSORTDATATYPE_FLOAT

To use the ColSortDataType property, you must create a multiple-column fpCombo or fpList control by setting the [Columns](#) property to a value greater than 0.

You can sort by the values in a column by specifying a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property and setting the [ColSorted](#) property.

Data Type

Integer (Enumerated)

See Also

[Sorting List Items](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColSorted](#), [ColSortSeq](#), [Columns](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColSorted Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the type of sort performed on a column in a multiple-column fpCombo or fpList control.

Syntax

C **UINT** LC_GetColSorted(**HWND** hWnd, **short FAR** *pValue);
UINT LC_SetColSorted(**HWND** hWnd, **short** value);

C++ **short Class::**GetColSorted(**void**);
Class::SetColSorted(**short** value);

Visual Basic [form.]control.ColSorted[= setting%]

Designer Page

[Sort designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - None (Default)	Does not sort list items	LC_COLSORTED_NONE
1 - Ascending	Sorts list items from the beginning of the alphabet or the lowest number	LC_COLSORTED_ASCENDING
2 - Descending	Sorts list items from the end of the alphabet or the highest number	LC_COLSORTED_DESCENDING

The ColSorted property is identical to the [Sorted](#) property but is used for multiple-column fpCombo or fpList controls. Use the [ColSortSeq](#) property to specify the sequence in which the column is sorted. Column sequence is zero-based, with 0 designating the first sort column, 1 designating the second sort column, and so on.

Before you set the ColSorted property, you must create a multiple-column fpCombo or fpList control with the [Columns](#) property and you must specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.

Data Type

Integer (Enumerated)

See Also

[Sorting List Items](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColSortSeq](#), [Columns](#), [Sorted](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColSortSeq Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the order in which a column in a multiple-column fpCombo or fpList control is sorted.

Syntax

C *UINT* LC_GetColSortSeq(*HWND* hWnd, *short FAR* *lpValue);
UINT LC_SetColSortSeq(*HWND* hWnd, *short* value);

C++ *short Class::*GetColSortSeq(*void*);
*Class::*SetColSortSeq(*short* value);

Visual Basic [form.]control.ColSortSeq[= value%]

Designer Page

[Sort designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the ColSortSeq property is 1, which indicates that no sorting is to occur on the column.

Before you set the ColSortSeq property, you must create a multiple-column fpCombo or fpList control with the [Columns](#) property. For each column, set either the [Col](#), [ColFromID](#), or [ColFromName](#) property to specify the column, the [ColSorted](#) property to specify the sort order, and the ColSortSeq property to specify the sequence in which the column is sorted. Column sequence is zero-based, with 0 designating the first sort column, 1 designating the second sort column, and so on.

Data Type

Integer

See Also

[Sorting List Items](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColSorted](#), [Columns](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColText Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the selected row's column text in a multiple-column fpCombo or fpList control.

Syntax

C **UINT** LC_GetColText(**HWND** hWnd, **LPSTR** buffer, **UINT** nBufferSize);
 UINT LC_SetColText(**HWND** hWnd, **LPCSTR** value);

C++ **LPSTR** **Class**::GetColText(**LPSTR** buffer, **UINT** bufferSize);
 Class::SetColText(**LPCSTR** value);

Visual Basic [form.]control.ColText[= text\$]

Remarks

Before you set the ColText property, you must create a multiple-column fpCombo or fpList control with the [Columns](#) property and you should specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.

Data Type

String

[Print](#)

[Copy](#)

[Close](#)

The following example creates a three-column simple combo style fpCombo control. The first column is displayed in the edit field and the third column is hidden. Text from the third column is displayed during the Change event.

C

```
void OnChangeLB(UNIT, int, Cwnd*, LPVOID)
{
    LC_SetCol(hWnd, 2);
    SetDlgItemText(hWndDLg, IDC_TEXT, LC_GetColText(hWnd);
}End Sub

LRESULT _export WINAPI WinMain(..... )
{
    LC_SetColumns(hWnd, 3);
    // Use simple combo style
    CbxSetStyle(hWnd, CBX_STYLE_SIMPLE_COMBO);
    // Display column 2 in edit field
    CbxSetColumnEdit(hWnd, 1);
    // Hide the third column
    LC_SetCol(hWnd, 2);
    LC_SetColHide(hWnd, TRUE);
}
```

C++

```
void CLBDialog::OnChangeLB(UNIT, int, Cwnd*, LPVOID)
{
    m_fpCombo->SetCol(2);
    SetDlgItemText(IDC_TEXT, m_fpCombo->GetColText( ));
}End Sub

Bool CLBDialog::OnInitDialog( )
{
    m_fpCombo->SetColumns(3);
    // Use simple combo style
    m_fpCombo->SetStyle(CBX_STYLE_SIMPLE_COMBO);
    // Display column 2 in edit field
    m_fpCombo->SetColumnEdit(1);
    // Hide the third column
    m_fpCombo->SetCol(2);
    m_fpCombo->SetColHide(TRUE);
}
```

Visual Basic

```
Sub fpCombo1_Change ( )
    fpCombo1.Col = 2
    Text1.Text = fpCombo1.ColText
End Sub

Sub Form_Load
    fpCombo1.Columns = 3
    ' Use simple combo style
    fpCombo1.Style = CBX_STYLE_SIMPLE_COMBO
    ' Display column 2 in edit field
    fpCombo1.ColumnEdit = 1
    ' Hide the third column
    fpCombo1.Col = 2
    fpCombo1.ColHide = True
End Sub
```

See Also

[Accessing List Items](#)

[Col](#), [ColFromID](#), [ColFromName](#), [Columns](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColumnBound Property

[See Also](#) [Example](#)

Applies To

fpCombo control

Description

Sets or returns the number of the column whose value is written to the database when a multiple-column fpCombo control is bound to a database field. This property is available for Visual Basic users only.

Syntax

Visual Basic `[form.]fpCombo1.ColumnBound[= value%]`

Designer Page

[Column subtab of the Data Binding designer page](#)

Remarks

The ColumnBound property designates which column's value is written to the database when the user selects an item in a multiple-column list and the Data control updates the record and saves the value.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the ColumnBound property is `1`, which writes all selected column values to the database.

You can create multiple columns in an fpCombo control by setting the [Columns](#) property. Each column can be bound to a database field by setting either the [Col](#), [ColFromID](#), or [ColFromName](#) property and the [ColDataField](#) property.

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

In the following example, the fpCombo control is bound to two different Data controls. To try this example, create an fpCombo control and two Data controls on a form. Bind the first Data control, Data1, to Visual Basic's BIBLIO.MDB database using the DatabaseName property and to the Titles table using the RecordSource property. Bind the fpCombo control to Data1 using the DataSource and [DataSourceList](#) properties. The list displays book titles and International Standard Book Numbers (ISBNs) per the following code.

Bind the second Data control, Data2, to the Orders table in a user-created database. The following code sets the DataField property to designate that Data2's ISBN field will receive data from the fpCombo control's edit field. The fpCombo control's edit field will read data from Data1's ISBN field per the [ColumnBound](#) property and display titles in the edit field per the [ColumnEdit](#) property.

Visual Basic

```
' Set combo box appearance
fpCombo1.EditHeight = 300
fpCombo1.ComboGap = 15
fpCombo1.ListLeftOffset = 100
fpCombo1.MaxDrop = 5
fpCombo1.Columns = 2
fpCombo1.GrayAreaColor = &H007D007D&

' Customize headers
fpCombo1.ColumnHeaderShow = True
fpCombo1.ColumnHeaderHeight = 400

' Bind combo box to database tables
' Display titles from Title table in edit field
fpCombo1.ColumnEdit = 0
' Write values from ISBN field in Title table
fpCombo1.ColumnBound = 3
' Write values to ISBN field in Order table
fpCombo1.DataField = "ISBN"

' Design columns in list
fpCombo1.DataAutoSizeCols = CBX_DATAAUTOSIZECOLS_MAXCOLWIDTH
fpCombo1.DataAutoHeadings = False
fpCombo1.ColumnWidthScale = LC_COLUMNWIDTHSCALE_PIXELS

' First column
fpCombo1.Col = 0
fpCombo1.ListApplyTo = LC_LISTAPPLYTO_SINGLE_COL_HEADER
fpCombo1.ColDataField = "#1"
fpCombo1.ColHeaderText = "Title of Book"
fpCombo1.AlignH = LC_ALIGNH_LEFT
fpCombo1.ColWidth = 400
fpCombo1.MultiLine = LC_MULTILINE_MULTIPLE_LINE

' Second column
fpCombo1.Col = 1
fpCombo1.ColDataField = "ISBN"
fpCombo1.ColHeaderText = "ISBN"
fpCombo1.ListApplyTo = LC_LISTAPPLYTO_SINGLE_COL_HEADER
fpCombo1.AlignH = LC_ALIGNH_CENTER
fpCombo1.ColWidth = 75
```

See Also

[Binding Columns to Fields in a Database](#)

[Col](#), [ColDataField](#), [ColFromID](#), [ColFromName](#), [ColumnEdit](#), [Columns DataSourceList](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColumnEdit Property

[See Also](#) [Example](#)

Applies To

fpCombo control

Description

Sets or returns whether one column value or all column values are displayed in the edit field of a multiple-column fpCombo control.

Syntax

C **UINT** CbxGetColumnEdit(**HWND** hWnd, **short FAR** *pValue);
UINT CbxSetColumnEdit(**HWND** hWnd, **short** value);

C++ **short** CfpComboBox::GetColumnEdit(**void**);
CfpComboBox::SetColumnEdit(**short** value);

Visual Basic [form.]fpCombo1.ColumnEdit[= value%]

Designer Page

[General subtab of the Columns designer page](#)
[Column subtab of the Data Binding designer page](#)

Remarks

The ColumnEdit property displays a single column value in the edit field of bound or nonbound fpCombo controls. Set the ColumnEdit property to the number of the list column to be displayed. The ColumnEdit property is zero-based; that is, setting the ColumnEdit property to 0 specifies that the first column values are displayed in the edit field.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the ColumnEdit property is

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

 1, which displays all column values in the edit field, with column separator characters between column values.

To display values in the edit field that are not displayed in the list, add the column of values to the list, and then hide the column with the [ColHide](#) property.

Data Type

Integer

See Also

[Binding Columns to Fields in a Database](#)

[ColHide](#), [Columns](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColumnHeaderHeight Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the height of the header in an fpCombo or fpList control.

Syntax

C **UINT** LC_GetColumnHeaderHeight(**HWND** hWnd, **long FAR ***pValue);
UINT LC_SetColumnHeaderHeight(**HWND** hWnd, **long** value);

C++ **long** **Class**::GetColumnHeaderHeight(**void**);
Class::SetColumnHeaderHeight(**long** value);

Visual Basic [form.]control.ColumnHeaderHeight[= value&]

Designer Page

[General subtab of the Columns designer page](#)

Remarks

You can display a header for a bound fpCombo or fpList control by setting the [ColumnHeaderShow](#) property to True.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the ColumnHeaderHeight property is 1, which bases the height of the header on the header font.

In Visual Basic, the measurement unit used by the ColumnHeaderHeight property depends on the setting of the form's ScaleMode property. The default ScaleMode setting is twips (1/1440 of an inch). Generally the ActiveX and VBX controls use twips as the default measurement unit, and the DLL control uses pixels as the default measurement unit.

Data Type

Integer (Long)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a four-column combo box control. The column header height and edit field height are 400 twips. The gap between the edit field and the drop-down arrow is 15 pixels. The gray area color is black and the maximum number of rows displayed is 5.

C

```
LC_SetColumns(hWnd, 4);
LC_SetColumnHeaderShow(hWnd, TRUE);
LC_SetColumnHeaderHeight(hWnd, 30);
Cbx_SetEditHeight(hWnd, 30);
Cbx_SetComboGap(hWnd, 15);
/* black RGB(0, 0, 0) */
Cbx_SetGrayAreaColor(hWnd, 0x00000000);
/* Display max of 5 rows */
Cbx_SetMaxDrop(hWnd, 5);
```

C++

```
fpCombo1->SetColumns(4);
fpCombo1->SetColumnHeaderShow(TRUE);
fpCombo1->SetColumnHeaderHeight(30);
fpCombo1->SetEditHeight(30);
fpCombo1->SetComboGap(15);
// black RGB(0, 0, 0)
fpCombo1->SetGrayAreaColor(0x00000000);
// Display max of 5 rows
fpCombo1->SetMaxDrop(5);
```

Visual Basic

```
fpCombo1.Columns = 4
fpCombo1.ColumnHeaderShow = True
fpCombo1.ColumnHeaderHeight = 400
fpCombo1.EditHeight = 400
fpCombo1.ComboGap = 15
' black RGB(0, 0, 0)
fpCombo1.GrayAreaColor = &H00000000&
' Display max of 5 rows
fpCombo1.MaxDrop = 5
```

See Also

[Customizing Column Headers](#)

[Wrapping Text in a List Pro Control](#)

[ColumnHeaderShow](#), [ColHeaderText](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColumnHeaderShow Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether to display the column header in an fpCombo or fpList control.

Syntax

C **UINT** LC_GetColumnHeaderShow(**HWND** hWnd, **BOOL FAR** *lpValue);
UINT LC_SetColumnHeaderShow(**HWND** hWnd, **BOOL** value);

C++ **BOOL** **Class**::GetColumnHeaderShow(**void**);
Class::SetColumnHeaderShow(**BOOL** value);

Visual Basic [form.]control.ColumnHeaderShow[= boolean%]

Designer Page

[General subtab of the Columns designer page](#)

Remarks

You can create a header for a bound fpCombo or fpList control by setting the ColumnHeaderShow property to True. If the [ColHeaderText](#) property does not specify a column header and the [DataAutoHeadings](#) property is set to True, the database field name is used.

The default value for the ColumnHeaderShow property is False, which specifies that headers are not displayed for multiple-column controls. To display a header for a single-column control, you must set the [Columns](#) property to 1 and the ColumnHeaderShow property to True.

For multiple-column fpCombo and fpList controls, setting the ColumnHeaderShow property to True displays a header for every column. If the column is bound to a database field and the DataAutoHeadings property is set to True, the data field name is the header text. To change the header text, set the ColHeaderText property.

Note You must display column headers to drag and drop columns.

Data Type

Integer (Boolean)

See Also

[Providing Column Headers](#)

[Binding Columns to Fields in a Database](#)

[ColHeaderText](#), [ColumnHeaderHeight](#), [Columns](#), [DataAutoHeadings](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColumnLevels Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the number of levels in every row of the control.

Syntax

C *UINT* LC_GetColumnLevels(*HWND* hWnd, *short FAR* *lpValue);
UINT LC_SetColumnLevels(*HWND* hWnd, *short* value);

C++ *short* *Class*::GetColumnLevels(*void*);
Class::SetColumnLevels(*short* value);

Visual Basic [form.]control.ColumnLevels[= value%]

Designer Page

[General subtab of the Columns designer page](#)

Remarks

The default value for the ColumnLevels property is 1.

Every row has the same number of levels. You can specify the level on which a column is displayed with the [ColLevel](#) property. You can set the height of a column in levels with the [ColLevelHeight](#) property.

If you display column headers and you have multiple levels in the control, the column headers also appear at multiple levels.

Note You cannot merge columns or rows if rows have multiple levels. If the number of levels is greater than zero, the [ColMerge](#) and [RowMerge](#) properties have no effect.

Data Type

Integer

See Also

[Creating Levels of Columns Within a Row](#)

[ColLevel](#), [ColLevelHeight](#), [ColMerge](#), [RowMerge](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Columns Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the number of columns to display in an fpCombo or fpList control.

Syntax

C	UINT LC_GetColumns(HWND hWnd, short FAR *lpValue); UINT LC_SetColumns(HWND hWnd, short value);
C++	short Class ::GetColumns(void); Class ::SetColumns(short value);
Visual Basic	[form.]control.Columns[= value%]

Designer Page

[Specific subtab of the Columns designer page](#)

Remarks

The default value for the Columns property is 0, which creates a single-column fpCombo or fpList control.

Setting the Columns property to 1 or greater creates a multiple-column fpCombo or fpList control. With a multiple-column fpCombo or fpList control, use the designated-column properties (such as [ColHide](#) and [ColWidth](#)) to customize each column. You can bind each column to a database field by setting the [ColDataField](#) property.

Though setting the Columns property to 1 creates an fpCombo or fpList control with only one column, this setting lets you use the designated-column properties to customize your list.

Note When the [WrapList](#) property is set to True, the fpCombo or fpList control wraps list items without creating columns. Therefore, no column properties apply, regardless of the Columns property setting.

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example creates a three-column fpCombo control. The list items are sorted by the first column, which is hidden.

C

```
/* Create three-column fpCombo control*/
LC_SetColumns(hWnd, 3);
/* Display contents of second column */
LC_SetColumnEdit(hWnd, 1);
/* Display headers */
LC_SetColumnHeaderShow(hWnd, TRUE);
/* First column hidden */
/* Primary sort */
LC_SetCol(hWnd, 0);
LC_SetColHeaderText(hWnd, "State");
LC_SetColSortDataType(hWnd, LC_COLSORTDATATYPE_TEXTNOCASE);
/* Make primary sort */
LC_SetColSortSeq(hWnd, 0);
LC_SetColSorted(hWnd, LC_COLSORTED_ASCENDING);
/* Hide column */
LC_SetColHide(hWnd, TRUE);
/* Second column */
/* Secondary sort */
LC_SetCol(hWnd, 1);
LC_SetColHeaderText(hWnd, "Company Name");
LC_SetColWidth(hWnd, 40);
LC_SetColSortDataType(hWnd, LC_COLSORTDATATYPE_TEXTNOCASE);
/* Make secondary sort */
LC_SetColSortSeq(hWnd, 1);
LC_SetColSorted(hWnd, LC_COLSORTED_ASCENDING);
LC_SetSortState(hWnd, LC_SORTSTATE_ACTIVE_RESORT);
/* Third column */
LC_SetCol(hWnd, 2);
LC_SetColHeaderText(hWnd, "Customer Name");
LC_SetColWidth(hWnd, 50);
```

C++

```
// Create three-column fpCombo control
fpCombo1->SetColumns(3);
// Display contents of second column
fpCombo1->SetColumnEdit(1);
// Display headers
fpCombo1->SetColumnHeaderShow(TRUE);
// First column hidden
// Primary sort
fpCombo1->SetCol(0);
fpCombo1->SetColHeaderText("State");
fpCombo1->SetColSortDataType(LC_COLSORTDATATYPE_TEXTNOCASE);
// Make primary sort
fpCombo1->SetColSortSeq(0);
fpCombo1->SetColSorted(LC_COLSORTED_ASCENDING);
// Hide column
fpCombo1->SetColHide(TRUE);
// Second column
// Secondary sort
fpCombo1->SetCol(1);
fpCombo1->SetColHeaderText("Company Name");
fpCombo1->SetColWidth(40);
fpCombo1->SetColSortDataType(LC_COLSORTDATATYPE_TEXTNOCASE);
// Make secondary sort
fpCombo1->SetColSortSeq(1);
fpCombo1->SetColSorted(LC_COLSORTED_ASCENDING);
fpCombo1->SetSortState(LC_SORTSTATE_ACTIVE_RESORT);
// Third column
```



```
fpCombo1->SetCol(2);  
fpCombo1->SetColHeaderText("Customer Name");  
fpCombo1->SetColWidth(50);
```

Visual Basic

```
' Create three-column fpCombo control  
fpCombo1.Columns = 3  
' Display contents of second column  
fpCombo1.ColumnEdit = 1  
' Display headers  
fpCombo1.ColumnHeaderShow = True  
' First column hidden  
' Primary sort  
fpCombo1.Col = 0  
fpCombo1.ColHeaderText = "State"  
fpCombo1.ColSortDataType = LC_COLSORTDATATYPE_TEXTNOCASE  
' Make primary sort  
fpCombo1.ColSortSeq = 0  
fpCombo1.ColSorted = LC_COLSORTED_ASCENDING  
' Hide column  
fpCombo1.ColHide = True  
' Second column  
' Secondary sort  
fpCombo1.Col = 1  
fpCombo1.ColHeaderText = "Company Name"  
fpCombo1.ColWidth = 40  
fpCombo1.ColSortDataType = LC_COLSORTDATATYPE_TEXTNOCASE  
' Make secondary sort  
fpCombo1.ColSortSeq = 1  
fpCombo1.ColSorted = LC_COLSORTED_ASCENDING  
fpCombo1.SortState = LC_SORTSTATE_ACTIVE_RESORT  
' Third column  
fpCombo1.Col = 2  
fpCombo1.ColHeaderText = "Customer Name"  
fpCombo1.ColWidth = 50
```

See Also

[Creating Multiple Columns](#)

[Col](#), [ColDataField](#), [ColHide](#), [ColWidth](#), [WrapList](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColumnSearch Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the number of the column searched when searching a multiple-column fpCombo or fpList control.

Syntax

C **UINT** LC_GetColumnSearch(**HWND** hWnd, **short FAR** *lpValue);
UINT LC_SetColumnSearch(**HWND** hWnd, **short** value);

C++ **short Class::**GetColumnSearch(**void**);
Class::SetColumnSearch(**short** value);

Visual Basic [form.]control.ColumnSearch[= value%]

Designer Page

[Search designer page](#)

Remarks

Use the ColumnSearch property to designate which column to search in a multiple-column fpCombo or fpList control. The ColumnSearch property is zero-based; that is, set the property to 0 to search the first column.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the ColumnSearch property is 1, which begins searching from the first column and searches every column.

To create a multiple-column fpCombo or fpList control, you must set the [Columns](#) property to a value greater than zero.

Data Type

Integer

[Print](#)

[Copy](#)

[Close](#)

The following example creates a two-column fpList control. To try this example, create an fpList control, a Text control, and a button control on a form. The fpList control searches for the search string entered into the Text control and scrolls to the first matching item.

C

```
LRESULT _export WINAPI WinMain(.....)
{
    /* Create a two-column fpList control */
    LC_SetColumns(hWnd, 2);
    /* Show column headers */
    LC_SetColumnHeaderShow(hWnd, TRUE);
    /* First column */
    LC_SetCol(hWnd, 0);
    LC_SetColHeaderText(hWnd, "City");
    LC_SetColWidth(hWnd, 60);
    /* Second column */
    LC_SetCol(hWnd, 1);
    LC_SetColHeaderText(hWnd, "Population");
    LC_SetRow(hWnd, -1);
    LC_SetInsertRow(hWnd, "Raleigh\t600,000");
    LC_SetInsertRow("Cary\t60,000");
    LC_SetInsertRow("Chapel Hill\t 45,000");
}

void OnClickm_LB(UNIT, int, Cwnd*, LPVOID)
{
    long Index;
    SetDlgItemText(IDC_BTN, "Find");
    /* Get the search string */
    LC_SetSearchText(hWnd, GetDlgItem(IDC_TEXT));
    /* Search the first column */
    LC_SetColumnSearch(hWnd, 0);
    /* Search for partial matches */
    LC_SetSearchMethod(hWnd, LC_SEARCHMETHOD_PARTIAL_MATCH);
    /* Start the search */
    LC_SetAction(hWnd, LC_ACTION_SEARCH);
    /* If a match is found, select the item */
    If LC_SetSearchIndex(hWnd) != -1
        LC_SetListIndex(hWnd, LC_GetSearchIndex(hWnd, &Index));
    Else
        /* If no match is found, reset the list */
        LC_SetListIndex(hWnd, 0);
}
```

C++

```
void CDlg::OnInitDialog( )
{
    // Create a two-column fpList control
    m_LB->SetColumns(2);
    // Show column headers
    m_LB->SetColumnHeaderShow(TRUE);
    // First column
    m_LB->SetCol(0);
    m_LB->SetColHeaderText("City");
    m_LB->SetColWidth(60);
    // Second column
    m_LB->SetCol(1);
    m_LB->SetColHeaderText("Population");
    m_LB->SetRow(-1);
    m_LB->SetInsertRow("Raleigh\t600,000");
    m_LB->SetInsertRow("Cary\t60,000");
    m_LB->SetInsertRow("Chapel Hill\t45,000");
}
```

```

void CDlg::OnClickm_LB(UNIT, int, Cwnd*, LPVOID)
{
    SetDlgItemText(IDC_BTN, "Find");
    // Get the search string
    GetDlgItemText(IDC_EDIT1, xdata, 100);
    m_LB->SetSearchText(xdata);
    // Search the first column
    m_LB->SetColumnSearch(0);
    // Search for partial matches
    m_LB->SetSearchMethod(LC_SEARCHMETHOD_PARTIAL_MATCH);
    // Start the search
    m_LB->SetAction(LC_ACTION_SEARCH);
    // If a match is found, select the item
    If (m_LB->SetSearchIndex X()! X = -1)
        m_LB->SetListIndex(m_LB->GetSearchIndex( ));
    Else
        // If no match is found, reset the list
        m_LB->SetListIndex(0);
}

```

Visual Basic

```

Sub Form_Load ()
' Create a two-column fpList control
fpList1.Columns = 2
' Show column headers
fpList1.ColumnHeaderShow = True
' First column
fpList1.Col = 0
fpList1.ColHeaderText = "City"
fpList1.ColWidth = 60
' Second column
fpList1.Col = 1
fpList1.ColHeaderText = "Population"
fpList1.Row = -1
fpList1.InsertRow = "Raleigh\t600,000"
fpList1.InsertRow = "Cary\t60,000"
fpList1.InsertRow = "Chapel Hill\t45,000"
End Sub

Sub Command1_Click ()
Command1.Caption = "Find"
' Get the search string
Form1.fpList1.SearchText = Text1.Text
' Search the first column
Form1.fpList1.ColumnSearch = 0
' Search for partial matches
Form1.fpList1.SearchMethod = LC_SEARCHMETHOD_PARTIAL_MATCH
' Start the search
Form1.fpList1.Action = LC_ACTION_SEARCH
' If a match is found, select the item
If Form1.fpList1.SearchIndex <> -1 Then
    Form1.fpList1.ListIndex = Form1.fpList1.SearchIndex
Else
    ' If no match is found, reset the list
    Form1.fpList1.ListIndex = 0
End If
End Sub

```

See Also

[Searching for List Items](#)

[AutoSearch](#), [Columns](#), [SearchIndex](#), [SearchMethod](#), [SearchText](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColumnSeparatorChar Property

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the character used to separate column values in a multiple-column fpCombo or fpList control.

Syntax

C **UINT** LC_GetColumnSeparatorChar(*HWND* hWnd, *short FAR* *pValue);
UINT LC_SetColumnSeparatorChar(*HWND* hWnd, *short* value);

C++ **short Class::**GetColumnSeparatorChar(*void*);
Class::SetColumnSeparatorChar(*short* value);

Visual Basic [form.]control.ColumnSeparatorChar[= value%]

Designer Page

[General subtab of the Columns designer page](#)

Remarks

The ColumnSeparatorChar property lets you define the character that separates the values in each column. The default value for the ColumnSeparatorChar property is 9 (the ASCII value of the Tab character).

In a multiple-column fpCombo control, the edit field displays a value from each column, separated by the character specified by the ColumnSeparatorChar property.

To specify the separator character, set the ColumnSeparatorChar property to a single ASCII character.

To separate values with . . .	Set ColumnSeparatorChar property to . . .
Tab character	9
Linefeed character	10
Carriage return	13
Space	32
Hyphen (-)	45
Forward slash (/)	47

You must insert this character when adding values to a multiple-column fpCombo or fpList control using the AddItem method or the [InsertRow](#) property.

To display only one column value in the edit field, set the [ColumnEdit](#) property.

Data Type

Integer

See Also

[Adding List Items](#)

[ColumnEdit](#), [InsertRow](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColumnWidthScale Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the measurement unit used to specify column and group widths in a multiple-column fpCombo or fpList control.

Syntax

C **UINT** LC_GetColumnWidthScale(**HWND** hWnd, **short FAR** *pValue);
UINT LC_SetColumnWidthScale(**HWND** hWnd, **short** value);

C++ **short Class::**GetColumnWidthScale(**void**);
Class::SetColumnWidthScale(**short** value);

Visual Basic [form.]control.ColumnWidthScale[= setting%]

Designer Page

[General subtab of the Columns designer page](#)

Remarks

The following settings are available:

Setting	Description	Constant
0 - Twips	Sets the measurement unit for column and group width to twips	LC_COLUMNWIDTHSCALE_TWIPS
1 - Pixels	Sets the measurement unit for column and group width to pixels	LC_COLUMNWIDTHSCALE_PIXELS
2 - Avg Char Width	(Default) Sets the measurement unit for column and group width to the average character width of the default font	LC_COLUMNWIDTHSCALE_AVG_CHAR_WIDTH
3 - Max Char Width	Sets the measurement unit for column and group width to the maximum character width of the default font	LC_COLUMNWIDTHSCALE_MAX_CHAR_WIDTH

Use the [ColWidth](#) property to specify the column width.

The [GrpWidth](#) property also uses the ColumnWidthScale property to specify the unit of measurement.

Data Type

Integer (Enumerated)

[Print](#)

[Copy](#)

[Close](#)

The following example creates a multiple-column fpList control and changes the background and text colors of the third column and row. Column widths are specified in twips.

C

```
LC_SetColumns(hWnd, 3);
LC_SetColumnHeaderShow(hWnd, FALSE);
LC_SetRow(hWnd, -1);
LC_SetInsertRow(hWnd, "1\tRaleigh\t919");
LC_SetInsertRow(hWnd, "2\tCharlotte\t704");
LC_SetInsertRow(hWnd, "3\tGreensboro\t910");
LC_SetInsertRow(hWnd, "4\tLouisburg\t919");
LC_SetInsertRow(hWnd, "5\tRoseboro\t910");
LC_SetColumnWidthScale(hWnd, LC_COLUMNWIDTHSCALE_TWIPS);
LC_SetCol(hWnd, 0);
LC_SetColWidth(hWnd, 300);
/* Set column colors */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SINGLE_ITEM);
LC_SetCol(hWnd, 2);
/* light blue RGB(0, 255, 255) */
LC_SetBackColor(hWnd, 0x00FFFF00);
/* dark blue RGB(0, 0, 128) */
LC_SetForeColor(hWnd, 0x00800000);
LC_SetRow(hWnd, 2);
/* Set row colors */
LC_SetListApplyTo(hWnd, LC_LISTAPPLYTO_SINGLE_ITEM);
/* blue RGB(0, 0, 255) */
LC_SetBackColor(hWnd, 0x00FF0000);
/* yellow RGB(255, 255, 0) */
LC_SetForeColor(hWnd, 0x0000FFFF);
/* Combine colors */
LC_SetHighestPrecedence(hWnd, LC_HIGHESTPRECEDENCE_COMBINED);
```

C++

```
fpList1->SetColumns(3);
fpList1->SetColumnHeaderShow(FALSE);
fpList1->SetRow(-1);
fpList1->SetInsertRow("1\tRaleigh\t919");
fpList1->SetInsertRow("2\tCharlotte\t704");
fpList1->SetInsertRow("3\tGreensboro\t910");
fpList1->SetInsertRow("4\tLouisburg\t919");
fpList1->SetInsertRow("5\tRoseboro\t910");
fpList1->SetColumnWidthScale(LC_COLUMNWIDTHSCALE_TWIPS);
fpList1->SetCol(0);
fpList1->SetColWidth(300);
// Set column colors
fpList1->SetCol(2);
fpList1->SetListApplyTo(LC_LISTAPPLYTO_SINGLE_ITEM);
// light blue RGB(0, 255, 255)
fpList1->SetBackColor(0x00FFFF00);
// dark blue RGB(0, 0, 128)
fpList1->SetForeColor(0x00800000);
// Set row colors
fpList1->SetRow(2);
fpList1->SetListApplyTo(LC_LISTAPPLYTO_SINGLE_ITEM);
// blue RGB(0, 0, 255)
fpList1->SetBackColor(0x00FF0000);
// yellow RGB(255, 255, 0)
fpList1->SetForeColor(0x0000FFFF);
// Combine colors
fpList1->SetHighestPrecedence(LC_HIGHESTPRECEDENCE_COMBINED);
```

Visual Basic

```
fpList1.Columns = 3
fpList1.ColumnHeaderShow = False
fpList1.Row = -1
fpList1.InsertRow = 1 & Chr$(9) & "Raleigh" & Chr$(9) & 919
fpList1.InsertRow = 2 & Chr$(9) & "Charlotte" & Chr$(9) & 704
fpList1.InsertRow = 3 & Chr$(9) & "Greensboro" & Chr$(9) & 910
fpList1.InsertRow = 4 & Chr$(9) & "Louisburg" & Chr$(9) & 919
fpList1.InsertRow = 5 & Chr$(9) & "Roseboro" & Chr$(9) & 910
fpList1.ColumnWidthScale = LC_COLUMNWIDTHSCALE_TWIPS
fpList1.Col = 0
fpList1.ColWidth = 300
' Set column colors
fpList1.Col = 2
fpList1.ListApplyTo = LC_LISTAPPLYTO_SINGLE_ITEM
' light blue RGB(0, 255, 255)
fpList1.BackColor = &H00FFFF00&
' dark blue RGB(0, 0, 128)
fpList1.ForeColor = &H00800000&
' Set row colors
fpList1.Row = 2
fpList1.ListApplyTo = LC_LISTAPPLYTO_SINGLE_ITEM
' blue RGB(0, 0, 255)
fpList1.BackColor = &H00FF0000&
' yellow RGB(255, 255, 0)
fpList1.ForeColor = &H0000FFFF&
' Combine colors
fpList1.HighestPrecedence = LC_HIGHESTPRECEDENCE_COMBINED
```

See Also

[Specifying the Column Width](#)

[Specifying the Group Width](#)

[ColWidth](#), [DataAutoSizeCols](#), [GrpWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ColWidth Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns the width of a column in an fpCombo or fpList control.

Syntax

C **UINT** LC_GetColWidth(**HWND** hWnd, **float FAR** *pValue);
UINT LC_SetColWidth(**HWND** hWnd, **float** value);

C++ **float** **Class**::GetColWidth(**void**);
Class::SetColWidth(**float** value);

Visual Basic [form.]control.ColWidth[= value&]

Designer Page

[Specific subtab of the Columns designer page](#)

Remarks

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The default value for the ColWidth property is 1, which is treated as 1 inch.

The width returned by the ColWidth property is based on the measurement unit set with the [ColumnWidthScale](#) property.

If you group columns, widths for columns on any given level are adjusted to fit within the parent group and the ColWidth property setting is ignored. For more information, see [Calculating the Width of Group Children](#).

Before you set the ColWidth property, you must specify a column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.

Before setting the ColWidth property for a bound fpCombo control or an fpList control with a bound list, set the [DataAutoSizeCols](#) property to 0 (Off).

If you do not specify the number of columns for the fpCombo or fpList control and the control is bound to a database table with multiple fields, the control determines the necessary number of columns to display in the list and sizes them using the DataAutoSizeCols property setting.

Data Type

Integer (Long)

See Also

[Specifying the Column Width](#)

[Calculating the Width of Group Children](#)

[Col](#), [ColFromID](#), [ColFromName](#), [ColumnWidthScale](#), [DataAutoSizeCols](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ComboGap Property

[See Also](#) [Example](#)

Applies To

fpCombo control

Description

Sets or returns the size in pixels of the gap between the edit field and the drop-down arrow in a drop-down combo style fpCombo control.

Syntax

C	UINT CbxGetComboGap(HWND hWnd, short FAR *pValue); UINT CbxSetComboGap(HWND hWnd, short value);
C++	short CfpComboBox::GetComboGap(void); CfpComboBox::SetComboGap(short value);
Visual Basic	[form.]fpCombo1.ComboGap[= value%]

Designer Page

[Misc subtab of the Appearance designer page](#)

Remarks

Drop-down combo style fpCombo controls display a gap between the edit field and the drop-down arrow. Use this property to set or return the size of the gap.

The default value for the ComboGap property is 7 pixels.

Keep the gap between the edit field and the drop-down arrow small to allow appropriate space for the edit field.

Data Type

Integer

See Also

[Choosing the fpCombo Control Style](#)

[ListWidth](#), [Style](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

DataAutoHeadings Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether the database field name is displayed as a header for a bound fpCombo or fpList control. This property is available for Visual Basic users only.

Syntax

Visual Basic `[form.]control.DataAutoHeadings[= boolean%]`

Designer Page

[General subtab of the Data Binding designer page](#)

Remarks

The default value for the DataAutoHeadings property is True. When set to True, the DataAutoHeadings property creates a header for the list and displays the database field name in the header.

When set to True, the DataAutoHeadings property displays a header even if the [ColumnHeaderShow](#) property is set to False.

Data Type

Integer (Boolean)

See Also

[Providing Column Headers](#)

[ColHeaderText](#), [ColumnHeaderShow](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

DataAutoSizeCols Property

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns whether columns in a multiple-column fpCombo or fpList control are automatically resized when bound to a database. This property is available for Visual Basic users only.

Syntax

Visual Basic `[form.]control.DataAutoSizeCols[= setting%]`

Designer Page

[General subtab of the Data Binding designer page](#)

Remarks

The DataAutoSizeCols property automatically sizes list columns based on the data in the database field to which they are bound. The following settings are available:

Setting	Description	Constant
0 - Off	Prevents columns from being sized automatically	LC_DATAAUTOSIZECOLS_OFF
1 - MaxColWidth	Resizes columns based on the maximum values in the bound database fields Note This setting requires slightly more processing time than other settings because the database table must be evaluated.	LC_DATAAUTOSIZECOLS_MAXCOLWIDTH
2 - BestGuess	(Default) Resizes columns based on the data type of the field	LC_DATAAUTOSIZECOLS_BESTGUESS
3 - HeaderWidth	Resizes columns based on the width of the column header	LC_DATAAUTOSIZECOLS_HEADERWIDTH

Consider setting the DataAutoSizeCols property to 3 (HeaderWidth) when you are working with data that contains few characters in each column but includes large headers for the columns. For example, numeric data may be short, but may need large headers to display information about the data.

If you want to set the size of each column separately at run time, set the DataAutoSizeCols property to 0 (Off) and set the [ColWidth](#) property to the width you want.

Data Type

Integer (Enumerated)

See Also

[Binding the Control to a Database](#)

[Binding Columns to Fields in a Database](#)

[ColDataField](#), [Columns](#), [ColWidth](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

DataBookmark Property

[See Also](#)

[Example](#)

Applies To

fpCombo, fpList controls

Description

Sets or returns a bookmark for a row in a bound fpCombo or fpList control. This property is available for Visual Basic users only.

Syntax

Visual Basic `[form.]control.DataBookmark[= text$]`

Remarks

To specify a bookmark with the DataBookmark property, you must specify a row with the [Row](#) property.

You can save the bookmark for the current record by assigning the value of the DataBookmark property to a variable. To return quickly to that record at any time after moving to a different record, set the recordset's Bookmark property to the value of that variable.

Data Type

String

[Print](#)

[Copy](#)

[Close](#)

To try the following example, create an fpList control and a Data control on a form. Bind the Data control to Visual Basic's BIBLIO.MDB database using the DatabaseName property and to the Authors table using the RecordSource property. Bind the fpList control to the Data control using the DataSource property. The following code sets the current record to be the fifth item in the list.

Visual Basic

```
Sub fpList1_DataLoaded ()  
    ' Set the current record to the record in row 5  
    fpList1.Row = 5  
    Data1.Recordset.Bookmark = fpList1.DataBookmark  
End Sub
```

See Also

[Row](#) property

Events

[Standard Visual Basic Events Supported by List Pro Controls](#)

[CloseUp](#)

[ColWidthChange](#)

[DataLoaded](#)

[DataRowChanged](#)

[DragDropCol](#)

[DragDropGrp](#)

[DropDown](#)

[GrpWidthChange](#)

[SelChange](#)

[TopChange](#)

[VirtualRequest](#)

[VirtualSearch](#)

Standard Visual Basic Events Supported by List Pro Controls

The following table lists the standard Visual Basic events that are supported by List Pro controls.

Refer to the Visual Basic documentation for more information on these events except for the DropDown and SelChange events. The List Pro online help provides additional information on these events.

Change
Click
DbClick
DragDrop
DragOver
[DropDown](#)
GotFocus
KeyDown
KeyPress
KeyUp
LinkClose
LinkError
LinkNotify
LinkOpen
LostFocus
MouseDown
MouseMove
MouseUp
[SelChange](#)

CloseUp Event

[See Also](#) [Example](#)

Applies To

fpCombo control

Description

Occurs when the list portion of a fpCombo control closes.

Syntax

C++ (MFC) `afx_msg void OnCloseUpfpCombo1(UINT, int, CWnd*, LPVOID);`
C++ (OWL) `void EvCloseUp(VBXEVENT far* event);`
Visual Basic `Sub fpCombo1_CloseUp ([Index As Integer])`

Parameters

The following parameter is available:

Parameter	Description
<i>Index</i>	Identifies a control if it is in an array

Remarks

This event does not occur when the [Style](#) property is set to 1 (Simple Combo).

When the user selects an item, the fpCombo control's list closes and the CloseUp event occurs. You can use the CloseUp event to make changes to a list (such as adding or removing items) after the user has selected an item.

[Print](#)

[Copy](#)

[Close](#)

The following example removes the selected item from the fpCombo when the fpCombo drop-down list closes.

```
Sub Form_Load ( )
    fpCombo1.AddItem "Pharr"
    fpCombo1.AddItem "Scarborough"
    fpCombo1.AddItem "Augusta"
    fpCombo1.AddItem "Cleveland"
    fpCombo1.AddItem "Brewton"
    fpCombo1.AddItem "Banner Elk"
    fpCombo1.AddItem "McMinnville"
    fpCombo1.AddItem "Minot"
    fpCombo1.AddItem "Steubenville"
End Sub

Sub fpCombo1_CloseUp ( )
    fpCombo1.RemoveItem fpCombo1.ListIndex
    fpCombo1.Text = ""
End Sub
```

See Also

[Style](#) property

[DropDown](#), [SelChange](#) events

ColWidthChange Event

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Occurs when the user changes the width of a column using the mouse.

Syntax

C++ (MFC) `afx_msg void OnColWidthChangecontrol(UINT, int, CWnd*, LPVOID);`

C++ (OWL) `void EvColWidthChange(VBXEVENT far* event);`

Visual Basic `Sub control_ColWidthChange ([Index As Integer,] Col As Integer)`

Parameters

The following parameter is available:

Parameter	Description
<i>Index</i>	Identifies a control if it is in an array
<i>Col</i>	The column whose width has been changed

Remarks

The ColWidthChange event occurs when the user releases the mouse button after changing the width of a column.

[Print](#)

[Copy](#)

[Close](#)

The following example creates a fpList control that lets users resize columns by dragging column or header borders. The example displays a message after each column width change that tells users which column changed and the new column width in twips.

```
Sub Form_Load ()
    fpList1.Columns = 2
    fpList1.AllowColResize = LC_ALLOWCOLRESIZE_RESIZECOLORHEADER
    fpList1.ColumnHeaderShow = True
    fpList1.Col = 0
    fpList1.ColHeaderText = "Place"
    fpList1.Col = 1
    fpList1.ColHeaderText = "State"
    fpList1.LineStyle = LC_LINESTYLE_FLAT
    fpList1.ScrollBarH = LC_SCROLLBAR_SHOW
    fpList1.ScrollBarV = LC_SCROLLBAR_SHOW_WHEN_NEEDED
    fpList1.AddItem "Ocracoke" & Chr(9) & "North Carolina"
    fpList1.AddItem "Cape Cod" & Chr(9) & "Massachusetts"
    fpList1.AddItem "Virginia Beach" & Chr(9) & "Virginia"
    fpList1.AddItem "Myrtle Beach" & Chr(9) & "South Carolina"
    fpList1.AddItem "Saint Simons" & Chr(9) & "Georgia"
    fpList1.AddItem "Daytona Beach" & Chr(9) & "Florida"
End Sub

Sub fpList1_ColWidthChange (Col As Integer)
    fpList1.ColumnWidthScale = LC_COLUMNWIDTHSCALE_TWIPS
    fpList1.Col = Col
    MsgBox "Column" & Col + 1 & " has been resized to " & fpList1.ColWidth & "
    twips."
End Sub
```

See Also

[AllowColResize](#) property

DataLoaded Event

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Occurs when the list is bound to a data control and the data has been loaded. This event is available only for Visual Basic users.

Syntax

Visual Basic Sub *control_DataLoaded* ([*Index* As **Integer**]

Parameters

The following parameter is available:

Parameter	Description
<i>Index</i>	Identifies a control if it is in an array

Remarks

When using a bound fpCombo or fpList control, perform any initializations in this event rather than the Form_Load event.

See Also

[DataSourceList](#) property

DataRowChanged Event

Applies To

fpCombo, fpList controls

Description

Occurs when the list is bound to a data control and a row of data has been changed. This event is available only for Visual Basic users.

Syntax

Visual Basic Sub *control_DataRowChanged*(*RowChanged* As **Long**)

Parameter

The following parameter is available:

Parameter	Description
<i>Index</i>	Identifies a control if it is in an array
<i>RowChanged</i>	Row number of the row that was changed

Remarks

When using a bound fpCombo or fpList control, perform any initializations in this event rather than the Form_Load event.

DragDropCol Event

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Occurs when the user has completed dragging a column to a new location.

Syntax

C++ (MFC) `afx_msg void OnDragDropColcontrol(UINT, int, CWnd*, LPVOID);`

C++ (OWL) `void EvDragDropCol(VBXEVENT far* event);`

Visual Basic `Sub control_DragDropCol ([Index As Integer,] Col As Integer,`

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

`NewCol As Integer, Cancel As Integer)`

Parameters

The following parameters are available:

Parameter	Description
<i>Index</i>	Identifies a control if it is in an array
<i>Col</i>	Returns the column that the user requested to move
<i>NewCol</i>	Returns the new location of the column that the user requested to move
<i>Cancel</i>	When set to True, cancels this operation

[Print](#) [Copy](#) [Close](#)

The following example creates a fpList control that lets users drag and drop columns to change their position in the control. The control displays a message when the column moves, telling the user which column moved and its new position.

```
Sub Form_Load ()
```

```

fpList1.Columns = 4
fpList1.AllowColDragDrop = LC_ALLOWCOLDRAGDROP_ALLCOLS
fpList1.ColumnHeaderShow = True
fpList1.Col = 0
fpList1.ColHeaderText = "State"
fpList1.Col = 1
fpList1.ColHeaderText = "Flower"
fpList1.ColWidth = 30
fpList1.Col = 2
fpList1.ColHeaderText = "Bird"
fpList1.ColWidth = 30
fpList1.Col = 3
fpList1.ColHeaderText = "Tree"
fpList1.ColWidth = 30
fpList1.AddItem "Alabama" & Chr(9) & "Camellia" & Chr(9) & "Yellowhammer" &
Chr(9) & "Southern Pine"
fpList1.AddItem "California" & Chr(9) & "Golden Poppy" & Chr(9) & "California
Valley Quail" & Chr(9) & "California Redwood"
fpList1.AddItem "Iowa" & Chr(9) & "Wild Rose" & Chr(9) & "Eastern Goldfinch" &
Chr(9) & "Oak"
fpList1.AddItem "Maine" & Chr(9) & "White Pine Cone and Tassel" & Chr(9) &
"Chickadee" & Chr(9) & "Eastern White Pine"
fpList1.AddItem "Minnesota" & Chr(9) & "Pink and White Lady's Slipper" & Chr(9) &
"Common Loon" & Chr(9) & "Red Pine"
fpList1.AddItem "Missouri" & Chr(9) & "Hawthorn" & Chr(9) & "Bluebird" & Chr(9) &
"Dogwood"
fpList1.AddItem "New Hampshire" & Chr(9) & "Purple Lilac" & Chr(9) & "Purple
Finch" & Chr(9) & "White Birch"
fpList1.AddItem "Wyoming" & Chr(9) & "Indian Paintbrush" & Chr(9) & "Meadowlark"
& Chr(9) & "Cottonwood"
End Sub

Sub fpList1_DragDropCol (Col As Integer, NewCol As Integer, Cancel As Integer)
MsgBox "Column " & Col+1 & " has been moved to column " & NewCol+1 & "."
End Sub

```

See Also

[AllowColDragDrop](#) property

DragDropGrp Event

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Occurs when the user has completed dragging a group to a new location.

Syntax

C++ (MFC) `afx_msg void OnDragDropGrpcontrol(UINT, int, CWnd*, LPVOID);`

C++ (OWL) `void EvDragDropGrp(VBXEVENT far* event);`

Visual Basic `Sub control_DragDropGrp ([Index As Integer,] Grp As Integer,`

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

`NewGrp As Integer, Cancel As Integer)`

Parameters

The following parameters are available:

Parameter	Description
<i>Index</i>	Identifies a control if it is in an array
<i>Grp</i>	Returns the group that the user requested to move
<i>NewGrp</i>	Returns the new location of the group that the user requested to move
<i>Cancel</i>	When set to True, cancels this operation

[Print](#) [Copy](#) [Close](#)

The following example creates a fpList control that lets users drag and drop groups to change their position in the control. The control displays a message when the group moves, telling the user which group moved and its new position.

```
Sub Form_Load ()
    fpList1.Columns = 4
    fpList1.Groups = 4
    fpList1.AllowGrpDragDrop = LC_ALLOWGRPDROP_ALLGRPS
    fpList1.GroupHeaderShow = True
    fpList1.Grp = 0
    fpList1.GrpHeaderText = "State"
    fpList1.Grp = 1
    fpList1.GrpHeaderText = "Flower"
    fpList1.GrpWidth = 20
    fpList1.Grp = 2
    fpList1.GrpHeaderText = "Bird"
    fpList1.GrpWidth = 20
    fpList1.Grp = 3
    fpList1.GrpHeaderText = "Tree"
    fpList1.GrpWidth = 20
    fpList1.Col = 0
```

```

fpList1.ColParentGroup = 0
fpList1.Col = 1
fpList1.ColParentGroup = 1
fpList1.Col = 2
fpList1.ColParentGroup = 2
fpList1.Col = 3
fpList1.ColParentGroup = 3
fpList1.AddItem "Alabama" & Chr(9) & "Camellia" & Chr(9) & "Yellowhammer" &
Chr(9) & "Southern Pine"
fpList1.AddItem "California" & Chr(9) & "Golden Poppy" & Chr(9) & "California
Valley Quail" & Chr(9) & "California Redwood"
fpList1.AddItem "Iowa" & Chr(9) & "Wild Rose" & Chr(9) & "Eastern Goldfinch" &
Chr(9) & "Oak"
fpList1.AddItem "Maine" & Chr(9) & "White Pine Cone and Tassel" & Chr(9) &
"Chickadee" & Chr(9) & "Eastern White Pine"
fpList1.AddItem "Minnesota" & Chr(9) & "Pink and White Lady's Slipper" & Chr(9) &
"Common Loon" & Chr(9) & "Red Pine"
fpList1.AddItem "Missouri" & Chr(9) & "Hawthorn" & Chr(9) & "Bluebird" & Chr(9) &
"Dogwood"
fpList1.AddItem "New Hampshire" & Chr(9) & "Purple Lilac" & Chr(9) & "Purple
Finch" & Chr(9) & "White Birch"
fpList1.AddItem "Wyoming" & Chr(9) & "Indian Paintbrush" & Chr(9) & "Meadowlark"
& Chr(9) & "Cottonwood"
End Sub

Sub fpList1_DragDropGrp (Grp As Integer, NewGrp As Integer, Cancel As Integer)
MsgBox "Group " & Grp+1 & " has been moved to group " & NewGrp +1 & "."
End Sub

```

See Also

[AllowGrpDragDrop](#) property

DropDown Event

[See Also](#)

Applies To

fpCombo control

Description

Occurs when the list portion of a fpCombo control is about to drop down.

Syntax

C++ (MFC) `afx_msg void OnDropDownfpCombo1(UINT, int, CWnd*, LPVOID);`

C++ (OWL) `void EvDropDown(VBXEVENT far* event);`

Visual Basic `Sub fpCombo1_DropDown ([Index As Integer,])`

Parameters

The following parameter is available:

Parameter	Description
<i>Index</i>	Identifies a control if it is in an array

Remarks

This event does not occur when the [Style](#) property is set to 1 (Simple Combo).

See Also

[Style](#) property

GrpWidthChange Event

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Occurs when the user changes the width of a group using the mouse.

Syntax

C++ (MFC) `afx_msg void OnGrpWidthChangecontrol(UINT, int, CWnd*, LPVOID);`
C++ (OWL) `void EvGrpWidthChange(VBXEVENT far* event);`
Visual Basic `Sub control_GrpWidthChange ([Index As Integer,] Grp As Integer)`

Parameters

The following parameter is available:

Parameter	Description
<i>Index</i>	Identifies a control if it is in an array
<i>Grp</i>	The group whose width has been changed

Remarks

The GrpWidthChange event occurs when the user releases the mouse button after changing the width of a group.

[Print](#)

[Copy](#)

[Close](#)

The following example creates a fpList control that lets users resize groups by dragging group or header borders. The example displays a message after each group width change that tells users which group changed and the new group width in twips.

```
Sub Form_Load ()
    fpList1.Groups = 2
    fpList1.Columns = 2
    fpList1.AllowGrpResize = LC_ALLOWGRPRESIZE_RESIZEGRPORHEADER
    fpList1.GroupHeaderShow = True
    fpList1.Grp = 0
    fpList1.GrpHeaderText = "Place"
    fpList1.Grp = 1
    fpList1.GrpHeaderText = "State"
    fpList1.Col = 0
    fpList1.ColParentGroup = 0
    fpList1.Col = 1
    fpList1.ColParentGroup = 1
    fpList1.AddItem "Ocracoke" & Chr(9) & "North Carolina"
    fpList1.AddItem "Cape Cod" & Chr(9) & "Massachusetts"
    fpList1.AddItem "Virginia Beach" & Chr(9) & "Virginia"
    fpList1.AddItem "Myrtle Beach" & Chr(9) & "South Carolina"
    fpList1.AddItem "Saint Simons" & Chr(9) & "Georgia"
    fpList1.AddItem "Daytona Beach" & Chr(9) & "Florida"
End Sub

Sub fpList1_GrpWidthChange (Grp As Integer)
    fpList1.ColumnWidthScale = LC_COLUMNWIDTHSCALE_TWIPS
    fpList1.Grp = Grp
    MsgBox "Grp " & Grp + 1 & " has been resized to " & fpList1.GrpWidth & " twips."
End Sub
```

See Also

[AllowGrpResize](#) property

SelChange Event

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Occurs when the user changes the selection in a fpCombo or fpList control.

Syntax

C++ (MFC) `afx_msg void OnSelChangecontrol(UINT, int, CWnd*, LPVOID);`
C++ (OWL) `void EvSelChange(VBXEVENT far* event);`
Visual Basic `Sub control_SelChange ([Index As Integer,] ItemIndex As Long)`

Parameters

The following parameter is available:

Parameter	Description
<i>Index</i>	Identifies a control if it is in an array
<i>ItemIndex</i>	Returns the index of the item that was just selected

Remarks

The SelChange event occurs when the same item is reselected or another item is selected. The selected item changes when the user clicks another list item or presses the arrow keys to highlight other items on the list. You can change the selected item at run time by setting the [ListIndex](#) property.

Use a SelChange event to update a list before the user closes the list. For the fpCombo control, the [CloseUp](#) event occurs before SelChange event.

[Print](#)

[Copy](#)

[Close](#)

The following example moves the current selection to the top of the fpList control.

```
Sub fpList1_SelChange(ItemIndex)
    fpList1.TopIndex = ItemIndex
End Sub
```

See Also

[Col](#), [ListIndex](#), [Row](#) properties

TopChange Event

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Occurs when the top row in the list changes, usually as a result of the user scrolling through the list using the mouse or the keyboard.

Syntax

C++ (MFC) `afx_msg void OnTopChangecontrol(UINT, int, CWnd*, LPVOID);`

C++ (OWL) `void EvTopChange(VBXEVENT far* event);`

Visual Basic `Sub control_TopChange ([Index As Integer,] OldTop As Long,`

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

`NewTop As Long)`

Parameters

The following parameters are available:

Parameter	Description
<i>Index</i>	Identifies a control if it is in an array
<i>OldTop</i>	Returns the row formerly displayed at the top of the list
<i>NewTop</i>	Returns the row currently displayed at the top of the list

[Print](#)

[Copy](#)

[Close](#)

The following example illustrates two fpList controls that change as the user scrolls through the first fpList control. To try this example, create a data control on your form and bind it to Visual Basic's BIBLIO.MDB database using the DatabaseName property. Bind it to the Titles table using the RecordSource property. Create two fpList controls on your form and bind them both to the data control with the DataSource property. Bind them both to the Title field with the DataField property.

```
Sub fpList1_TopChange (OldTop As Long, NewTop As Long)
    fpList2.TopIndex = NewTop
End Sub
```


See Also

[TopIndex](#) property

VirtualRequest Event

[See Also](#)

Applies To

fpCombo, fpList controls

Description

Occurs when the list receives a request, such as when the user wants to scroll through the list, and the list is in virtual mode.

Syntax

C++ (MFC) `afx_msg void OnVirtualRequestcontrol(UINT, int, CWnd*, LPVOID);`
 C++ (OWL) `void EvVirtualRequest(VBXEVENT far* event);`
 Visual Basic `Sub control_VirtualRequest ([Index As Integer,] ActionRequested As Integer,`

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

`RowFirst As Long, RowCount As Long, Pos As Long, Ret As Long)`

Parameters

The following parameters are available:

Parameter	Description																				
<i>Index</i>	Identifies a control if it is in an array																				
<i>ActionRequested</i>	Returns information about the action requested																				
	<table border="1"> <thead> <tr> <th>Action</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0 - Is Home</td> <td>If the first record is currently in memory, set the <i>Ret</i> parameter to True or False.</td> </tr> <tr> <td>1 - Is End</td> <td>If the last record is currently in memory, set the <i>Ret</i> parameter to True or False.</td> </tr> <tr> <td>2 - Down</td> <td>Fill the list with the next set of records (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i>. Set <i>Ret</i> to the number of records actually loaded.)</td> </tr> <tr> <td>3 - Up</td> <td>Fill the list with the previous set of records (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i>. Set <i>Ret</i> to the number of records actually loaded.)</td> </tr> <tr> <td>4 - Home</td> <td>Fill the list, starting with the first record in the database (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i>. Set <i>Ret</i> to the number of records actually loaded.)</td> </tr> <tr> <td>5 - End</td> <td>Fill the list, starting with the last record in the database (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i>. Set <i>Ret</i> to the number of records actually loaded.)</td> </tr> <tr> <td>6 - Refresh</td> <td>Refresh the current records (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i>. Set <i>Ret</i> to the number of records actually loaded.)</td> </tr> <tr> <td>7 - Rows Removed</td> <td>Remove from memory the number of records designated by <i>RowCount</i>, starting at <i>RowFirst</i></td> </tr> <tr> <td>8 - Pos</td> <td>Move the scroll box to the position specified by <i>Pos</i></td> </tr> </tbody> </table>	Action	Description	0 - Is Home	If the first record is currently in memory, set the <i>Ret</i> parameter to True or False.	1 - Is End	If the last record is currently in memory, set the <i>Ret</i> parameter to True or False.	2 - Down	Fill the list with the next set of records (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i> . Set <i>Ret</i> to the number of records actually loaded.)	3 - Up	Fill the list with the previous set of records (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i> . Set <i>Ret</i> to the number of records actually loaded.)	4 - Home	Fill the list, starting with the first record in the database (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i> . Set <i>Ret</i> to the number of records actually loaded.)	5 - End	Fill the list, starting with the last record in the database (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i> . Set <i>Ret</i> to the number of records actually loaded.)	6 - Refresh	Refresh the current records (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i> . Set <i>Ret</i> to the number of records actually loaded.)	7 - Rows Removed	Remove from memory the number of records designated by <i>RowCount</i> , starting at <i>RowFirst</i>	8 - Pos	Move the scroll box to the position specified by <i>Pos</i>
Action	Description																				
0 - Is Home	If the first record is currently in memory, set the <i>Ret</i> parameter to True or False.																				
1 - Is End	If the last record is currently in memory, set the <i>Ret</i> parameter to True or False.																				
2 - Down	Fill the list with the next set of records (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i> . Set <i>Ret</i> to the number of records actually loaded.)																				
3 - Up	Fill the list with the previous set of records (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i> . Set <i>Ret</i> to the number of records actually loaded.)																				
4 - Home	Fill the list, starting with the first record in the database (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i> . Set <i>Ret</i> to the number of records actually loaded.)																				
5 - End	Fill the list, starting with the last record in the database (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i> . Set <i>Ret</i> to the number of records actually loaded.)																				
6 - Refresh	Refresh the current records (The number of records designated by <i>RowCount</i> are requested, starting at <i>RowFirst</i> . Set <i>Ret</i> to the number of records actually loaded.)																				
7 - Rows Removed	Remove from memory the number of records designated by <i>RowCount</i> , starting at <i>RowFirst</i>																				
8 - Pos	Move the scroll box to the position specified by <i>Pos</i>																				

(The number of records designated by *RowCount* are requested, starting at *RowFirst*. Set *Ret* to the number of records actually loaded.)

RowFirst Designates the first row in the list to fill with data
(When *ActionRequested* is 3 (Up) or 5 (End), start filling the list at $RowFirst+RowCount-1$)

RowCount Designates the number of rows to load into the list

Pos Designates the new position of the scroll box when *ActionRequested* is 8 (Pos)

Ret Used to return a value for each action designated by *ActionRequested*
(This value should be filled before exiting the event. Action 7 (Rows Removed) is the only action that does not need a return value.)

Remarks

The request made of the list usually requires the control to load more rows. If the list is bound to a data control, your application need not respond to this event.

See Also

[VirtualSearch](#) event

[VirtualMode](#) property

VirtualSearch Event

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Occurs when the fpCombo or fpList control is in virtual mode and the user requests a search. This property enables the user to search the entire database.

Syntax

C++ (MFC) `afx_msg void OnVirtualSearchcontrol(UINT, int, CWnd*, LPVOID);`

C++ (OWL) `void EvVirtualSearch(VBXEVENT far* event);`

Visual Basic `Sub control_VirtualSearch ([Index As Integer,] SearchString As String,`

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

`SearchType As Integer, RowFound As Long, Ret As Integer)`

Remarks

Because the list in a fpCombo or fpList control only looks at the data in memory, it cannot accurately search the entire data base. This event allows the user to search the entire database.

The following arguments are provided:

Parameter	Description	Description
<i>Index</i>	Identifies a control if it is in an array	
<i>SearchString</i>	Designates the string for which to search (This value can consist of one or more characters.)	
<i>SearchType</i>	Designates the type of search requested The following two values may be combined with the Or operator: Value 1 - Search Method Greater 2 - Search Method Partial	Description The found string may be greater than or equal to the search string (If the value does not contain a 1, the found string must equal the search string.) Requests a partial search (If the value does not contain a 2, the found string must exactly match the search string.)
<i>RowFound</i>	Returns the row number of the found string (The control attempts to find the requested string in memory. If found, this argument contains the row number of the found text; otherwise, its value is -1. If you want the control to select the row found in memory, return from this event. If you do not want the control to select the found row but rather to search the rest of the database, set this parameter to -1 before returning.)	
<i>Ret</i>	Designates the results of the search request (Set this parameter to True if the request was successful;	

otherwise, set it to False.)

Note In Visual Basic, when the fpCombo or fpList control is bound, Visual Basic custom controls cannot directly communicate to the access engine to perform a search. Therefore, you must request the search at run time in Visual Basic code. Refer to the example for code you might want to place in the VirtualSearch event.

[Print](#)

[Copy](#)

[Close](#)

The following example shows code you can place in the VirtualSearch event for a bound fpList control if you want to search the entire database.

```
Sub fpList1_VirtualSearch (SearchString As String, SearchType As Integer, RowFound
As Long, Ret As Integer)
  If RowFound = -1 Then
    ' If Greater than
    If SearchType And 2 Then
      Data1.RecordSet.FindFirst "Author >= '" & SearchString & "'"
    ' If Partial match
    ElseIf SearchType And 1 Then
      Data1.RecordSet.FindFirst "Author Like '" & SearchString & "*'"
    ' Search for whole match
    Else
      Data1.RecordSet.FindFirst "Author = '" & SearchString & "'"
    End If
    Ret = Not Data1.RecordSet.NoMatch
  End If
End Sub
```

See Also

[VirtualRequest](#) event

[VirtualMode](#) property

Functions and Methods

The following topics describe the available List Pro functions for VBX users and methods for OCX users.

[LP_GetCellPos](#)

[LP_GetMaxSize](#)

[ListPro_GetControlhWnd](#)

LP_GetCellPos Function

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Returns the cell position of a cell if it is currently visible on the screen.

Syntax

C **BOOL** LP_GetCellPos(**HWND** hWnd, **short** Col, **long** Row, **LPRECT** lpRect);

C++ **BOOL** LP_GetCellPos(**LPCVOID** hWnd, **short** Col, **long** Row, **LPLONG** lpx,

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

LPLONG lpy, **LPLONG** lpWidth, **LPLONG** lpHeight, **short** wUnits);

Visual Basic Function LP_GetCellPos(hWnd As **Control**, ByVal nCol As **Integer**,

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ByVal iRow As **Long**, lpx As **Long**, lpy As **Long**,

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

lpWidth As **Long**, lpHeight As **Long**, ByVal Units As **Integer**)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

As **Integer**

Parameters

The following parameters are available:

DLL Parameters

Parameter	Description
<i>hWnd</i>	Window handle of the List Pro control
<i>Col</i>	Column number of cell for which to return rectangle coordinates
<i>Row</i>	Row number of cell for which to return rectangle coordinates
<i>lpRect</i>	Pointer to variable that receives the rectangle coordinates

C++ Parameters

Parameter	Description
<i>hWnd</i>	List Pro control handle
<i>Col</i>	Column number
<i>Row</i>	Row number
<i>lpx</i>	Pointer to x-position
<i>lpy</i>	Pointer to y-position
<i>lpWidth</i>	Pointer to column width
<i>lpHeight</i>	Pointer to row height
<i>wUnits</i>	Unit of measurement. If 0 is used, the unit is twips.

Use the following:

Value	Constant	Description
1	FP_CELLPOSUNITS_PIXELS	Pixels

ActiveX, VBX Parameters

Parameter	Description
<i>hCtlList</i>	List Pro control handle
<i>nCol</i>	Column number
<i>iRow</i>	Row number

lpx x-position
lpy y-position
lpWidth Column width
lpHeight Row height
Units Unit of measurement. If 0 is used, the unit is twips.
Use the following:

Value	Constant	Description
1	FP_CELLPOSUNITS_PIXELS	Pixels

Return Value

DLL: TRUE if the function completes successfully; FALSE otherwise.

[Print](#)

[Copy](#)

[Close](#)

The following example inserts five rows in to a combo box control. When the user clicks the command button, the x-position, y-position, width, height, and unit of measurement is printed.

C

```
LPRECT xy;
Cwnd *ctrl;
HWND hWnd;

    ctrl = GetDlgItem(IDC_COMBO); // button1
    hWnd = ctrl->m_hWnd;

LP_GetCellPos(hWnd,0,1,xy);
```

C++

```
LPRECT xy;
Cwnd *ctrl;
HWND hWnd;

    ctrl = GetDlgItem(IDC_COMBO); // button1
    hWnd = ctrl->m_hWnd;

LP_GetCellPos(hWnd,0,1,xy);
```

Visual Basic (OCX)

```
Private Sub Form_Load()
    fpCombo1.Row = -1
    fpCombo1.InsertRow = 1 & Chr$(9) & "Raleigh" & Chr$(9) & 919
    fpCombo1.InsertRow = 2 & Chr$(9) & "Charlotte" & Chr$(9) & 704
    fpCombo1.InsertRow = 3 & Chr$(9) & "Greensboro" & Chr$(9) & 910
    fpCombo1.InsertRow = 4 & Chr$(9) & "Louisburg" & Chr$(9) & 919
    fpCombo1.InsertRow = 5 & Chr$(9) & "Roseboro" & Chr$(9) & 910
End Sub

Private Sub Command2_Click()
    Dim xpos As Long
    Dim ypos As Long
    Dim w As Long
    Dim h As Long
    Dim u As Integer

    fpCombo1.GetCellPos 1, 1, xpos, ypos, w, h, u

    Debug.Print xpos, ypos, w, h, u
End Sub
```

Visual Basic (VBX)

```
Private Sub Form_Load()
    fpCombo1.Row = -1
    fpCombo1.InsertRow = 1 & Chr$(9) & "Raleigh" & Chr$(9) & 919
    fpCombo1.InsertRow = 2 & Chr$(9) & "Charlotte" & Chr$(9) & 704
    fpCombo1.InsertRow = 3 & Chr$(9) & "Greensboro" & Chr$(9) & 910
    fpCombo1.InsertRow = 4 & Chr$(9) & "Louisburg" & Chr$(9) & 919
    fpCombo1.InsertRow = 5 & Chr$(9) & "Roseboro" & Chr$(9) & 910
End Sub

Private Sub Command2_Click()
    Dim xpos As Long
    Dim ypos As Long
    Dim w As Long
    Dim h As Long
    Dim u As Integer
```

```
Dim ret As Integer  
ret = LP_GetCellPos(fpComb01, 1, 1, xpos, ypos, w, h, u)  
Debug.Print xpos, ypos, w, h, u  
End Sub
```

See Also

[Col](#), [ColWidth](#), [Row](#), [RowHeight](#) properties

LP_GetMaxSize Function

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Returns the maximum column width or row height based on the current text, current picture, or both.

Syntax

C

```
long LP_GetMaxSize(HWND hWnd, short nCol, long IRow,
    

|    |    |    |    |    |
|----|----|----|----|----|
| PD | RD | WR | RT | DT |
| ✓  | ✓  | ✓  | ✓  | ✓  |

short wMethod, LPRECT lpRect);
```

C++

```
long LP_GetMaxSize(LPVOID hWnd, short nCol, long IRow,
    

|    |    |    |    |    |
|----|----|----|----|----|
| PD | RD | WR | RT | DT |
| ✓  | ✓  | ✓  | ✓  | ✓  |

short wMethod, short wUnits);
```

Visual Basic Function LP_GetMaxSize(hCtlList As **Control**, ByVal nCol As **Integer**,

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

 ByVal IRow As **Long**, ByVal Method As **Integer**,

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

 ByVal Units As **Integer**) As **Long**

Parameters

The following parameters are available:

DLL Parameters

Parameter	Description
<i>hWnd</i>	List Pro control handle
<i>nCol</i>	

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To return row height, set to 1.
 Otherwise, set to specific column.

IRow

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To return column width, set to 1.
 Otherwise, set to a specific row.

wMethod

Calculation method
 Use one of the following values:

Value	Description
FP_MAXSIZEMETHOD_BASIC	Row height: Returns height of single line of text Column width: Returns the width of the three-dimensional cell border
FP_MAXSIZEMETHOD_TEXT	Row height: Returns height of text, taking into account multiple lines, if displayed Column width: Returns width of single line of text, ignoring multiple lines, if displayed
FP_MAXSIZEMETHOD_PICT	Returns row height or column width based on graphic only.
FP_MAXSIZEMETHOD_TEXTPICT	Row height: Returns height of text and graphic, taking into account multiple lines, if

Row height: Returns height of single line of text
Column width: Returns the width of the three-dimensional cell border
Row height: Returns height of text, taking into account multiple lines, if displayed
Column width: Returns width of single line of text, ignoring multiple lines, if displayed
 Returns row height or column width based on graphic only.
Row height: Returns height of text and graphic, taking into account multiple lines, if

displayed

Column width: Returns width of single line of text and graphic, ignoring multiple lines, if displayed

lpRect Pointer to variable that receives the rectangle coordinates

C++ Parameters

Parameter **Description**
hWnd List Pro control handle
nCol

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To return row height, set to 1.
Otherwise, set to specific column.

IRow

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To return column width, set to 1.
Otherwise, set to a specific row.

wMethod Calculation method
Use one of the following values:

Value

FP_MAXSIZEMETHOD_BASIC

FP_MAXSIZEMETHOD_TEXT

FP_MAXSIZEMETHOD_PICT

FP_MAXSIZEMETHOD_TEXTPICT

Description

Row height: Returns height of single line of text

Column width: Returns the width of the three-dimensional cell border

Row height: Returns height of text, taking into account multiple lines, if displayed

Column width: Returns width of single line of text, ignoring multiple lines, if displayed

Returns row height or column width based on graphic only.

Row height: Returns height of text and graphic, taking into account multiple lines, if displayed

Column width: Returns width of single line of text and graphic, ignoring multiple lines, if displayed

wUnits Unit of measurement
Using the OR operator, combine the following values. Note that a value of 0 assumes twips.

Value	Constant	Description
1	FP_MAXSIZEUNITS_PIXELS	Returns value in pixels
2	FP_MAXSIZEUNITS_ROW	Returns the row height instead of the column width

ActiveX, VBX Parameters

Parameter **Description**
hCtrlList List Pro control handle
nCol

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To return row height, set to 1.
Otherwise, set to specific column.

IRow

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To return column width, set to 1.
Otherwise, set to a specific row.

Method Calculation method
Use one of the following values:

Value
FP_MAXSIZEMETHOD_BASIC

Description

Row height: Returns height of single line of text

Column width: Returns the width of the three-dimensional

FP_MAXSIZEMETHOD_TEXT	cell border Row height: Returns height of text, taking into account multiple lines, if displayed Column width: Returns width of single line of text, ignoring multiple lines, if displayed
FP_MAXSIZEMETHOD_PICT	Returns row height or column width based on graphic only.
FP_MAXSIZEMETHOD_TEXTPICT	Row height: Returns height of text and graphic, taking into account multiple lines, if displayed Column width: Returns width of single line of text and graphic, ignoring multiple lines, if displayed

Units

Unit of measurement
Using the OR operator, combine the following values. Note that a value of 0 assumes twips.

Value	Constant	Description
1	FP_MAXSIZEUNITS_PIXELS	Returns value in pixels
2	FP_MAXSIZEUNITS_ROW	Returns the row height instead of the column width

Remarks

You use the LP_GetMaxSize function to return either the maximum height of a specified row or the maximum width of a specified column, or the maximum width or height of a cell.

Return Type

Integer (Long)

[Print](#)

[Copy](#)

[Close](#)

The following example returns the column width of a list box control using the basic method. If the maximum column width returned is 0, the column width is set to the minimum desired width of X pixels.

C++

```
long ret;
char buffer[20];

ret = LP_GetMaxSize(hwnd,1,-1,FP_MAXSIZEMETHOD_TEXT,1);

if (ret = 0)
{
    sprintf(buffer, "%f", ret);
    MessageBox(buffer, "Reset ColWidth to 12", MB_OK);
    LC_SetCol(hwnd, 1);
    LC_SetColWidth(hwnd, 12);
}
```

C++

```
long ret;
char buffer[20];

ret = LP_GetMaxSize(hwnd,1,-1,FP_MAXSIZEMETHOD_TEXT,1);

if (ret = 0)
{
    sprintf(buffer, "%f", ret);
    MessageBox(buffer, "Reset ColWidth to 12", MB_OK);
    fpList1->SetCol(hwnd, 1);
    fpList1->SetColWidth(hwnd, 12);
}
```

Visual Basic (OCX)

```
Dim ret As Long

ret = fpList1.GetMaxSize(1, -1, FP_MAXSIZEMETHOD_TEXT, 1)
Debug.Print ret
If ret = 0 Then
    MsgBox "Reset ColWidth to 12"
    fpList1.Col = 1
    fpList1.ColWidth = 12
End If
```

Visual Basic (VBX)

```
Dim ret As Long

ret = LP_GetMaxSize(fpList1, 1, -1, FP_MAXSIZEMETHOD_TEXT, 1)
Debug.Print ret
If ret = 0 Then
    MsgBox "Reset ColWidth to 12"
    fpList1.Col = 1
    fpList1.ColWidth = 12
End If
```

See Also

[ColWidth](#), [RowHeight](#) properties

ListPro_GetControlhWnd Function

[See Also](#) [Example](#)

Applies To

fpCombo, fpList controls

Description

Determine the window handle of any Visual Basic control. This function is available for VBX controls only.

Syntax

Visual Basic Function ListPro_GetControlhWnd (*hCtl* As **Control**) As **Integer**

Description

Use these functions to determine the window handle of any Visual Basic control. You can use these functions with the [DataSourcehWnd](#) property to bind a List Pro control to a data control on another form. If the data control is on another form, be sure to specify the form name with the data control name (Form2.Data1).

The ListPro_GetControlhWnd function is defined in your LP.BAS file. Add the file to your project to use the function.

Return Type

The window handle (integer)

[Print](#)

[Copy](#)

[Close](#)

The following example retrieves the window handle of the data control, Data1, which is located on Form2:

```
fpList1.DataSourceehWnd = ListPro_GetControlhWnd(Form2.Data1)
```

See Also

[DataSourceWnd](#), [DataSourceWndList](#) properties

Structures

LB_KEYDOWN

LB_VIRTUALREQUEST

LB_VIRTUALSEARCH

LB_KEYDOWN Structure

Applies To

fpList control

Definition

```
typedef struct tagLB_KEYDOWN
{
    WORD wKeyCode;
    WORD wShiftState;
} LB_KEYDOWN, FAR *LPLB_KEYDOWN;
```

Fields

The LB_KEYDOWN structure has the following fields:

Field	Description
<i>wKeyCode</i>	A key code
<i>wShiftState</i>	The state of the Shift, Ctrl, and Alt keys at the time of the event (The <i>Shift</i> parameter is a bit field, with the least-significant bits corresponding to the Shift key (bit 0), the Ctrl key (bit 1), and the Alt key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both Ctrl and Alt are pressed, the value of <i>Shift</i> is 6.)

Remarks

This structure is used with the [LBM_KEYDOWN](#) message to inform the application that the user has just pressed a key while the control has the focus.

LB_VIRTUALREQUEST Structure

Definition

```
typedef struct tagLB_VIRTUALREQUEST
{
    WORD wActionRequested;
    long lRowFirst;
    long lRowCount;
    long lPos;
} LB_VIRTUALREQUEST, FAR *LPLB_VIRTUALREQUEST;
```

Fields

The LB_VIRTUALREQUEST structure has the following fields:

Field	Description		Description
<i>wActionRequested</i>	Use one of the following values:	Value	
		LB_VIRTACTION_ISTOP	The top row in memory
		LB_VIRTACTION_ISEND	The last row in memory
		LB_VIRTACTION_DOWN	The next group of data is needed.
		LB_VIRTACTION_UP	The previous group of data is needed.
		LB_VIRTACTION_HOME	The first group of data is needed.
		LB_VIRTACTION_END	The last group of data is needed.
		LB_VIRTACTION_REFRESH	Used if the contents of the buffer are being refreshed.
		LB_VIRTACTION_ROWSREMOVED	
		LB_VIRTACTION_POS	
<i>lRowFirst</i>	Virtual top row at which reading begins		
<i>lRowCount</i>	Number of rows requested		
<i>lPos</i>	Scroll bar position		

Remarks

This structure is used with the [LBM_VIRTUALREQUEST](#) message to inform the application that the control needs more data when the control is in virtual mode.

LP_VIRTUALSEARCH Structure

Definition

```
typedef struct tagLB_VIRTUALSEARCH
{
    LPSTR lpszSearchString;
    WORD  wSearchType;
    long  lRowFound;
} LB_VIRTUALSEARCH, FAR *LPLB_VIRTUALSEARCH;
```

Remarks

Fields

The LB_VIRTUALSEARCH structure has the following fields:

Field	Description		Description
<i>lpszSearchString</i>	Pointer to search string		
<i>wSearchType</i>	Type of search requested		
	Value		
	1 - Search Method Greater		The found string may be greater than or equal to the search string (If the value does not contain a 1, the found string must equal the search string.)
	2 - Search Method Partial		Requests a partial search (If the value does not contain a 2, the found string must exactly match the search string.)
<i>lRowFound</i>	Number of row where match was found		

Remarks

This structure is used with the [LBM_VIRTUALSEARCH](#) message to inform the application a search request is initiated and the control is in virtual mode.

Messages

Note For the fpCombo or fpList control, you must include the LBS_NOTIFY style in the RC file to receive all notifications.

[Overview](#)

[Windows Messages Supported by List Pro Controls](#)

[CBM_KEYDOWN](#)

[CBM_KEYPRESS](#)

[LBM_CLICK](#)

[LBM_COLWIDTHCHANGE](#)

[LBM_DRAGDROP](#)

[LBM_DRAGDROPGRP](#)

[LBM_GRPWIDTHCHANGE](#)

[LBM_KEYDOWN](#)

[LBM_KEYPRESS](#)

[LBM_VIRTUALREQUEST](#)

[LBM_VIRTUALSEARCH](#)

[LBN_TOPITEM](#)

Overview: Messages

Use messages to communicate with the List Pro DLL controls. As in the Windows-supplied messages, data is passed to the control with the *wParam* and *lParam* function parameters. Use the SendMessage API function to send messages to controls.

Note For the fpCombo or fpList control, you must include the LBS_NOTIFY style in the RC file to receive all notifications.

Windows Messages Supported by List Pro Controls

The following lists contain all Windows messages for combo box and list box controls that are supported by List Pro controls.

- [fpCombo Control](#)
- [fpList Control](#)

fpCombo Control

The following Windows messages for combo boxes are supported by the fpCombo control. For more information on these messages, see your Windows documentation.

CB_ADDSTRING
CB_DELETESTRING
CB_FINDSTRING
CB_FINDSTRINGEXACT
CB_GETCOUNT
CB_GETCURSEL
CB_GETDROPPEDSTATE
CB_GETEDITSEL
CB_GETITEMDATA
CB_GETITEMHEIGHT
CB_GETLBTEXT
CB_GETLBTEXTLEN
CB_INSERTSTRING
CB_LIMITTEXT
CB_RESETCONTENT
CB_SELECTSTRING
CB_SETCURSEL
CB_SETEDITSEL
CB_SETITEMDATA
CB_SETITEMHEIGHT
CB_SHOWDROPDOWN
CBN_CLOSEUP
CBN_DROPDOWN
CBN_EDITCHANGE
CBN_ERRSPACE
CBN_KILLFOCUS
CBN_SELCHANGE
CBN_SETFOCUS

fpList Control

The following Windows messages for list boxes are supported by the fpList control. For more information on these messages, see your Windows documentation.

LB_ADDSTRING
LB_DELETESTRING
LB_FINDSTRING
LB_FINDSTRINGEXACT
LB_GETCARETINDEX
LB_GETCOUNT
LB_GETCURSEL
LB_GETHORIZONTALEXTENT
LB_GETITEMDATA
LB_GETITEMHEIGHT
LB_GETITEMRECT

LB_GETSEL
LB_GETSELCOUNT
LB_GETSELITEMS
LB_GETTEXT
LB_GETTEXTLEN
LB_GETTOPINDEX
LB_INSERTSTRING
LB_RESETCONTENT
LB-SELECTSTRING
LB_SETCARETINDEX
LB_SETCOLUMNWIDTH
LB_SETCURSEL
LB_SETHORIZONTALEXTENT
LB_SETITEMDATA
LB_SETITEMHEIGHT
LB_SETITEMRANGE
LB_SETSEL
LB_SETTABSTOPS
LB_SETTOPINDEX
LBN_DBLCLK
LBN_ERRSPACE
LBN_KILLFOCUS
LBN_SELCHANGE
LBN_SETFOCUS

CBM_KEYDOWN Message

Description

Sent to the owner of the fpCombo control when the user presses (KeyDown) a key while the control has the focus.

Parameters

This message has the following parameters:

Parameter	Description
<i>wParam</i>	The fpCombo control window handle
<i>lParam</i>	Pointer to the LB_KEYDOWN structure

Return Value

None

CBM_KEYPRESS Message

Description

Sent to the owner of the fpCombo control when the user releases (KeyUp) a key while the control has the focus.

Parameters

This message has the following parameters:

Parameter	Description
<i>wParam</i>	The fpCombo control window handle
<i>lParam</i>	LPWORD (Pointer to ASCII character that was pressed)

Return Value

None

LBM_CLICK Message

Description

Sent to the owner of the fpList control each time the user pressed the left mouse button.

Parameters

This message has the following parameters:

Parameter	Description
<i>wParam</i>	The fpList control window handle
<i>lParam</i>	Not used

Return Value

None

LBM_COLWIDTHCHANGE Message

Description

Sent to the owner of the fpList or fpCombo control when the width of column changes.

Parameters

This message has the following parameters:

Parameter	Description
<i>wParam</i>	The fpList of fpCombo control window handle
<i>lParam</i>	Column number of column whose width has changed

Return Value

None

LBM_DRAGDROPCOL Message

Description

Sent to the owner of the fpList or fpCombo control when the user drags and drops a column.

Parameters

This message has the following parameters:

Parameter	Description
<i>wParam</i>	The fpList or fpCombo control window handle
<i>lParam</i>	LOWORD Column number of column being moved HIWORD New position number of column that was moved

Return Value

A non-zero return value cancels the drag-drop request.

LBM_DRAGDROPGRP Message

Description

Sent to the owner of the fpList or fpCombo control when the user drags and drops a group.

Parameters

This message has the following parameters:

Parameter	Description
<i>wParam</i>	The fpList or fpCombo control window handle
<i>lParam</i>	LOWORD Group number of group being moved HIWORD New position number of group that was moved

Return Value

A non-zero return value cancels the drag-drop request.

LBM_GRPWIDTHCHANGE Message

Description

Sent to the owner of the fpList or fpCombo control when the width of group changes.

Parameters

This message has the following parameters:

Parameter	Description
<i>wParam</i>	The fpList or fpCombo control window handle
<i>lParam</i>	Group number of group whose width has changed

Return Value

None

LBM_KEYDOWN Message

Description

Sent to the owner of the fpList control when the user presses (KeyDown) a key while the control has the focus.

Parameters

This message has the following parameters:

Parameter	Description
<i>wParam</i>	The fpList control window handle
<i>lParam</i>	Pointer to the LB_KEYDOWN structure

Return Value

None

LBM_KEYPRESS Message

Description

Sent to the owner of the fpList control when the user releases (KeyUp) a key while the control has the focus.

Parameters

This message has the following parameters:

Parameter	Description
<i>wParam</i>	The fpList control window handle
<i>lParam</i>	LPWORD (Pointer to ASCII character that was pressed)

Return Value

None

LBM_VIRTUALREQUEST Message

Description

Sent to the owner of the fpList control when the control needs more data when the control is in virtual mode.

Parameters

This message has the following parameters:

Parameter	Description
<i>wParam</i>	The fpList control window handle
<i>lParam</i>	Pointer to the LB_VIRTUALREQUEST structure

Remarks

Refer to the [VirtualRequest](#) event for more information.

Return Value

Number of rows loaded

LBM_VIRTUALSEARCH Message

Description

Sent to the owner of the fpList control when a search request is initiated and the control is in virtual mode.

Parameters

This message has the following parameters:

Parameter	Description
<i>wParam</i>	The fpList control window handle
<i>lParam</i>	Pointer to the LB_VIRTUALSEARCH structure

Remarks

Refer to the [VirtualSearch](#) event for more information.

Return Value

Returns True if successful

LBN_TOPITEM Message

Description

Sent to the owner of the fpList control when the user scrolls the list in the vertical direction. The owner receives this notification through a WM_COMMAND message from the control.

Return Value

None

Styles

[Overview](#)

[Windows Styles Supported by List Pro Controls](#)

[CBS_DESCENDINGORDER](#)

[CBS_SEARCHEXACTCASE](#)

[CBS_THREEDIN](#)

[CBS_THREEDOUT](#)

[LBS_DESCENDINGORDER](#)

[LBS_SEARCHEXACTCASE](#)

[LBS_THREEDIN](#)

[LBS_THREEDOUT](#)

Overview: Styles

Styles are settings that DLL users can apply to a control when they create it. You can assign a defined value or a numeric value for a style. Click one of the listed styles for a brief description and its values.

Windows Styles Supported by List Pro Controls

The following lists contain all Windows styles for combo box and list box controls that are supported by List Pro controls.

- [fpCombo Control](#)
- [fpList Control](#)

fpCombo Control

The following Windows styles for combo boxes are supported by the fpCombo control. For more information on these styles, see your Windows documentation.

CBS_AUTOHSCROLL
CBS_DROPDOWN
CBS_DROPDOWNLIST
CBS_NOINTEGRALHEIGHT
CBS_SIMPLE
CBS_SORT

fpList Control

The following Windows styles for list boxes are supported by the fpList control. For more information on these styles, see your Windows documentation.

LBS_EXTENDSEL
LBS_MULTICOLUMN
LBS_MULTIPLESEL
LBS_NOINTEGRALHEIGHT
LBS_NOREDRAW
LBS_NOTIFY*
LBS_SORT
LBS_USETABSTOPS

* For the fpList control, you must include the LBS_NOTIFY style in the RC file to receive all notifications.

CBS_DESCENDINGORDER

If sorting is specified, sorts items in an fpCombo control in descending order.

Constant Value

CB_DESCENDINGORDER

CBS_SEARCHEXACTCASE

Searches using exact case in an fpCombo control.

Constant Value

CB_SEARCHEXACTCASE

CBS_THREEDIN

The fpCombo control displays a lowered three-dimensional appearance.

Constant Value

CB_THREEDIN

CBS_THREEDOUT

The fpCombo control displays a raised three-dimensional appearance.

Constant Value

CB_THREEDOUT

LBS_DESCENDINGORDER

If sorting is specified, sorts items in an fpList control in descending order.

Constant Value

LB_DESCENDINGORDER

LBS_SEARCHEXACTCASE

Searches using exact case in an fpList control.

Constant Value

LB_SEARCHEXACTCASE

LBS_THREEDIN

The fpList control displays a lowered three-dimensional appearance.

Constant Value

LB_THREEDIN

LBS_THREEDOUT

The fpList control displays a raised three-dimensional appearance.

Constant Value

LB_THREEDOUT

Introducing List Pro

- [System Requirements](#)
- [Upgrading from Aware/VBX](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Redistributing List Pro Controls](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Important Information for C and C++ Users](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Documentation Conventions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Contacting FarPoint Technologies, Inc.](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Comment Form](#)

System Requirements

To run List Pro™, you need a computer running Microsoft® Windows® version 3.1 or later or Windows NT version 3.5 or later, running in standard or enhanced mode. You also need a development environment that supports ActiveX, DLL, or VBX controls.

Upgrading from Aware/VBX

Instructions for upgrading from our Aware/VBX™ product are included in the file UPGRAD.WRI that accompanies this product. For a list of the files provided in List Pro version 2.0, refer to the file INSTALL.WRI that accompanies this product.

If you have used our Aware/VBX product, review the information provided in the topic [Information for Users of Previous Products](#).

Redistributing List Pro Controls

Information about redistributing List Pro controls is provided in the file REDIST.WRI that accompanies this product. For a list of the files provided in List Pro version 2.0, refer to the file INSTALL.WRI that accompanies this product.

Documentation Conventions

List Pro documentation uses the following conventions:

Example	Description
Illustrations	Illustrations of the Input Pro controls and other screen objects were created in the Windows 95™ operating environment unless otherwise noted.
Example	Choose this item to display an example window. Note Pixel values in examples are calculated for a 640 x 480 screen resolution.
ActiveX	Text indicates the condition for the subsequent instructions or explanation. For example, "ActiveX" indicates that the subsequent information pertains only to the ActiveX control. Possible conditions include version of the control, programming environment, and run-time versus design-time interaction.
A: \SETUP MYFILE.TXT <i>value, color</i>	Words you need to type appear in this font. Words in all capital letters indicate file and path names. Italicized items in programming syntax are placeholders for information you supply.
short, BOOL	In programming syntax, bold, italicized arguments indicate data types, pointers, and user-defined types.
Class	In programming syntax, blue text indicates additional information is provided in a pop-up window. To access the pop-up window, click the blue text. (Italicized blue text indicates variable content, as described in the pop-up window.)
[= <i>value</i> &]	In programming syntax, items inside square brackets are optional. This symbol indicates a reference to the how-to-guides. This symbol occurs in the tutorial.



PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

In programming syntax, the line continuation character indicates that code continued from one line to the next in the documentation should be typed all on one line in the code window.

Important Information for C and C++ Developers

Throughout the online help, discussions of setting properties are intended to cover setting characteristics of the control using functions. Generally, there is a correspondence between each property of the control and the corresponding functions, as described in [Getting and Setting Property Values](#). Therefore, if you use the DLL, refer to the appropriate property reference page in the online [Reference Guide](#) for information about performing your task using functions.

You may also find the illustrative examples throughout the online help useful. Where applicable, each example in the online help provides code showing how to perform a task in C, C++, and Visual Basic.

Contacting FarPoint Technologies, Inc.

If you discover a problem with either the software or the accompanying online help, or if you would like to share your thoughts about this product with FarPoint Technologies, please send us a completed [Comment Form](#).

Send us the Comment Form using one of the following methods, or if you call, please provide the information requested on the form to our Technical Support representative. For additional assistance, contact our Technical Support department.

To contact our Technical Support department

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Call our Technical Support department by phone at (919) 460-1887.

FarPoint Technologies Technical Support department is available between the hours of 9:00 a.m. and 5:30 p.m. eastern time.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Fax us at (919) 460-7606.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Visit our **World Wide Web** site at <http://www.fpoint.com>.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Send **e-mail** to us at

- fpsupport@fpoint.com (technical support)
- farpoint@fpoint.com (general mail)
- fpsales@fpoint.com (sales)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Contact the FarPoint forum on **CompuServe**® (type GO FARPOINT and choose Library 12).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Contact our **FTP** site (<ftp.fpoint.com> and change to the directory /fpoint.com)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Write us at:

FarPoint Technologies, Inc.
133 Southcenter Court
Suite 1000
Morrisville, NC 27560

List Pro Version 2.0 Comment Form

If you encounter a problem with this product, or if you would like to share your comments with FarPoint Technologies, please complete and send this form to FarPoint Technologies using one of the methods described in [Contacting FarPoint Technologies, Inc.](#)

If you are reporting a problem, please include a detailed description of the problem along with the steps necessary to reproduce it. The best way to convey the problem is to send us a sample application that reproduces the problem. If you have a sample, please upload it to our CompuServe forum or to our Internet address. Be sure to include the name of the ZIP file* that you uploaded.

Today's date: _____ Customer name: _____

Company name: _____ Phone number: (____) _____

Fax number: (____) _____ e-mail address: _____

*Sample application file name: _____

Control or controls used in application (circle all that apply):

fpCombo fpList

Version Number: _____ Control Date: _____

Control Type (circle one or more): 16-bit DLL 32-bit DLL VBX 32-bit ActiveX

Development environment: _____ Version: _____

(for example, Visual Basic, Visual C++, Borland C++)

Description of the problem:

Steps necessary to reproduce the problem:

Comments/Documentation suggestions:

Information for Users of Previous Products

If you have used our Aware/VBX product, you should read the following topics to familiarize yourself with the updates available in this new product, List Pro. For additional information and instructions on upgrading existing projects, consult the UPGRAD.WRI file that accompanies this product.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Column Designer](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Changes to Existing Properties](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[New Properties](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[New Events](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Changes to Functions](#)

Column Designer

The Column Designer has been replaced by the [FarPoint Property Designer](#). The FarPoint Property Designer provides a quick and simple way for you to design an fpCombo or fpList control. You can view the run-time appearance of the control as you create it.

The [List Pro Tutorial](#) shows how to create a list box using the FarPoint Property Designer.

Changes to Existing Properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Changes to Property Names](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[OnFocusShadow Property](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Column Properties](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Header Properties](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Line Properties](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Row Properties](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Selected Item Properties](#)

Changes to Property Names

The properties listed in the following table have been renamed, but operate similarly to the Aware/VBX properties. The original property names are still supported for backward compatibility. However, the old properties might not be supported in future versions. It is recommended that you replace them.

Former name	New name
ColLock	<u>ColLockResize</u>
DropShadowColor	<u>BorderDropShadowColor</u>
DropShadowWidth	<u>BorderDropShadowWidth</u>
HeaderHeight	<u>ColumnHeaderHeight</u>
HeaderShow	<u>ColumnHeaderShow</u>
OnFocusInvert3D	<u>ThreeDOnFocusInvert</u>

The OnFocusShadow property has been renamed and the new property operates differently than the Aware/VBX property. The OnFocusShadow property is still supported for backward compatibility.

OnFocusShadow Property

The OnFocusShadow property has been changed to the [BorderDropShadow](#) property. The former OnFocusShadow property was a boolean property; the new BorderDropShadow property is an enumerated property. The settings listed in the following table are available for the BorderDropShadow property.

Setting	Description
0 - None	No drop shadow is displayed
1 - Always	Drop shadow is always displayed
2 - On Focus	Drop shadow is displayed when the control has the focus

The OnFocusShadow property is still supported for backward compatibility.

Column Properties

The following column properties are now obsolete, but are retained for backward compatibility.

Property	Replaced by
ColAlignH	ListApplyTo + AlignH
ColBackColor	ListApplyTo + BackColor
ColForeColor	ListApplyTo + ForeColor
ColHeaderAlignH	ListApplyTo + AlignH
ColMultiLine	ListApplyTo + MultiLine
ColPicture	ListApplyTo + Picture
ColPictureAlignH	ListApplyTo + PictureAlignH
ColPictureSel	ListApplyTo + PictureSel

Header Properties

The following header properties are now obsolete, but are retained for backward compatibility.

Property	Replaced by
Header3DStyle	ListApplyTo + LineStyle
Header3DText	ListApplyTo + List3DText
Header3DTextHighlightColor	ListApplyTo + List3DTextHighlightColor
Header3DTextShadowColor	ListApplyTo + List3DTextShadowColor
Header3DWidth	ListApplyTo + ListWidth
HeaderBackColor	ListApplyTo + BackColor
HeaderFontBold	ListApplyTo + FontBold
HeaderFontItalic	ListApplyTo + FontItalic
HeaderFontName	ListApplyTo + FontName
HeaderFontSize	ListApplyTo + FontSize
HeaderFontStrike	ListApplyTo + FontStrikethru
HeaderFontUnder	ListApplyTo + FontUnderline
HeaderForeColor	ListApplyTo + ForeColor

Line Properties

The following line properties are now obsolete, but are retained for backward compatibility.

Property	Replaced by
LineHighlightColor	LineApplyTo + Line3DLight
LineShadowColor	LineApplyTo + Line3DDark
LineSolidColor	LineApplyTo + LineColor
LineStyleH	LineApplyTo + LineStyle
LineStyleV	LineApplyTo + LineStyle

Row Properties

The following row properties are now obsolete, but are retained for backward compatibility.

Property	Replaced by
RowBackColor	ListApplyTo + BackColor
RowForeColor	ListApplyTo + ForeColor
RowPicture	ListApplyTo + Picture
RowPictureAlignH	ListApplyTo + PictureAlignH
RowPictureCol	ListApplyTo + Picture
RowPictureSel	ListApplyTo + PictureSel

Selected Item Properties

The following selected item properties are now obsolete, but are retained for backward compatibility.

Property	Replaced by
Sel3DStyle	ListApplyTo + LineApplyTo
Sel3DWidth	ListApplyTo + Line3DWidth

SelBackColor	ListApplyTo + BackColor
SelForeColor	ListApplyTo + ForeColor
SelHighlightColor	ListApplyTo + Line3DLight
SelShadowColor	ListApplyTo + Line3DDark

New Properties

The following table lists the new properties that have been added for the List Pro controls.

Property	Description
Action (9 - Virtual Refresh)	Forces the control to discard the current page of data and re-request the data currently in the buffer
Action (10 - Insert Group)	Inserts a group before the specified group
Action (11 - Delete Group)	Deletes a group
Action (12 - Clone Column)	Clones (copies all attributes except data, identification number or name, and data field setting) a specified column and inserts the duplicate column to the right of the specified column
AlignH	Specifies horizontal alignment of text
AlignV	Specifies vertical alignment of text
AllowGrpDragDrop	Enables the user to move groups by dragging and dropping
AllowGrpResize	Enables the user to resize groups by dragging borders
Appearance	Specifies a predefined border style for the control
BorderGrayAreaColor	Specifies the gray area color of the surrounding border to distinguish it from the forms background color
ColFromID	Specifies the column you want to work with by ID (identifier number)
ColFromName	Specifies the column you want to work with by name
ColID	Specifies the column ID (identifier number)
ColLevel	Determines the level within a row on which a column is displayed
ColLevelHeight	Specifies the height of a column in levels
ColMerge	Specifies whether the control merges adjacent cells in a column that contain the same text
ColName	Specifies the column name
ColParentGroup	Determines the parent group of a specified column
ColPos	Specifies the position number of a column within the control
ColPosInParent	Specifies the position number of a column within its parent
ColumnLevels	Determines the number of levels in every row of the control
ExtendCol	Determines whether columns are displayed beyond the last row containing data in a multiple-column control
ExtendRow	Determines whether rows and row selection are displayed beyond the last column containing data in a multiple-column control
FontEmpty	Clears the existing font properties so that the area chosen can inherit the font attributes of its hierarchical predecessor
GroupHeaderHeight	Specifies the height of the group header
GroupHeaderShow	Determines whether the group header is displayed
Groups	Specifies the number of groups in the control
Grp	Specifies the group you want to work with
GrpFromID	Specifies the group you want to work with by ID (identifier number)
GrpFromName	Specifies the group you want to work with by name
GrpHeaderText	Determines the text to display in the specified group header
GrpHide	Hides a specified group and its children
GrpID	Specifies the group ID (identifier number)
GrpLockResize	Prevents the user from resizing a specified group
GrpName	Specifies the group name
GrpParentGroup	Specifies the parent group of a specified group
GrpPos	Specifies the position number of a group within the control
GrpPosInParent	Specifies the position number of a group within its parent
GrpsFrozen	Specifies the number of leftmost groups that do not scroll across horizontally
GrpWidth	Specifies the width of a group
JoinID	Specifies the identification number for a joined set of cells

Line3DDark	Specifies the shadow color of three-dimensional lines
Line3DLight	Specifies the highlight color of three-dimensional lines
Line3DWidth	Determines the width of three-dimensional lines in pixels
LineApplyTo	Determines whether line properties apply to the lines between rows, between columns, between columns when multiple levels exist, or to all lines
LineColor	Specifies the color of flat lines between columns and rows
LineStyle	Specifies the appearance of lines between ListApplyTo items
ListApplyTo	Specifies where to apply designated-list properties
ListGrayAreaColor	Determines the color of the gray area between cells
MaxEditLen	Specifies the maximum number of characters that can be typed in the edit field of a combo box
MergeAdjustView	Specifies whether the view of cell contents of merged rows or columns is adjusted as you scroll through the list
MouseOverArea	Returns the part of the combo box control over which the mouse is positioned
MouseOverGrp	Returns the number of the group over which the mouse is positioned
MouseOverGrpHeader	Returns the number of the group header over which the mouse is positioned
MultiLine_Property	Determines whether list item text displays in a single line or in multiple lines
NextSel	Determines the row number of the next selected row
PictureAlignH	Horizontally aligns the graphic
PictureAlignV	Vertically aligns the graphic
PictureSel	Specifies the graphic displayed when a list item is selected
RowMerge	Determines whether the control merges adjacent cells in a row that contain the same text
TextOrientation	Determines whether the text and graphics are rotated 90, 180, or 270 degrees or whether the text is displayed vertically
ThreeDFrameColor	Specifies the color of the three-dimensional frame

New Events

The following table lists the new events that have been added for the List Pro controls.

Event	Description
DataRowChanged	Occurs when the user changes the data in a row
DragDropGroup	Occurs when the user has completed dragging a group to a new location
GrpWidthChange	Occurs when the user changes the width of a group using the mouse

Changes to Functions

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Changes to Function Names](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[New Functions](#)

Changes to Function Names

The AWARELC_GetControlhWnd function has been renamed [ListPro_GetControlhWnd](#). The original function name is still supported for backward compatibility.

New Functions

The following table lists the new functions that have been added for the List Pro controls.

Function	Description
<u>LP_GetCellPos</u>	Returns the x- and y-coordinates of the cell position
<u>LP_GetMaxSize</u>	Determines the maximum column width of a specific column or the maximum row height of a specific row

List Pro Controls

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Features](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Components](#)

List Pro Features

The List Pro controls offer unique capabilities that are described throughout the online help. The following topics describe some of List Pros most often used features, including features new to List Pro. For more information about using these features, consult the topics listed.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Overview](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[DLL Controls](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[ActiveX Controls](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[FarPoint Property Designer](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[List Pro Tutorial](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Combo Box Styles](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Columns, Rows, and Cells](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Groups](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Headers](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Text and Graphics](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Appearance](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[List Items](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Data Binding](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Virtual Mode](#)

Overview

The fpCombo and fpList controls display large amounts of data in a scrolling list. Additionally, the fpCombo control can provide an edit field for data entry. Both controls can display up to two billion list items and can handle more than 64 kilobytes of data, which is the limit of the Visual Basic combo box and list box. The fpCombo and fpList controls also bind to the Visual Basic Data control.

fpCombo control (drop-down combo box)

The diagram shows an fpCombo control with an edit field at the top containing the number '1'. Below it is a list of five rows. The first row is selected and highlighted in blue. Labels on the left point to the edit field, header, selected item, and a row.

Au_ID	Author	Year Born
1	Adams, Pat	
2	Adrian, Merv	
3	Ageloff, Roy	1943
4	Andersen, Virginia	
5	Antonovich Michael P.	

fpList control

The diagram shows an fpList control with a scrolling list of text items. The first item is selected and highlighted in blue. A label on the left points to the scrolling list.

Database management: developing application systems us
Select-- SQL ; the relational database language
dBase IV programming
Step-by-step dBase IV
Guide to ORACLE
The database experts' guide to SQL
Oracle/SQL: a professional programmer's guide
SQL 400: A Professional Programmer's Guide

DLL Controls

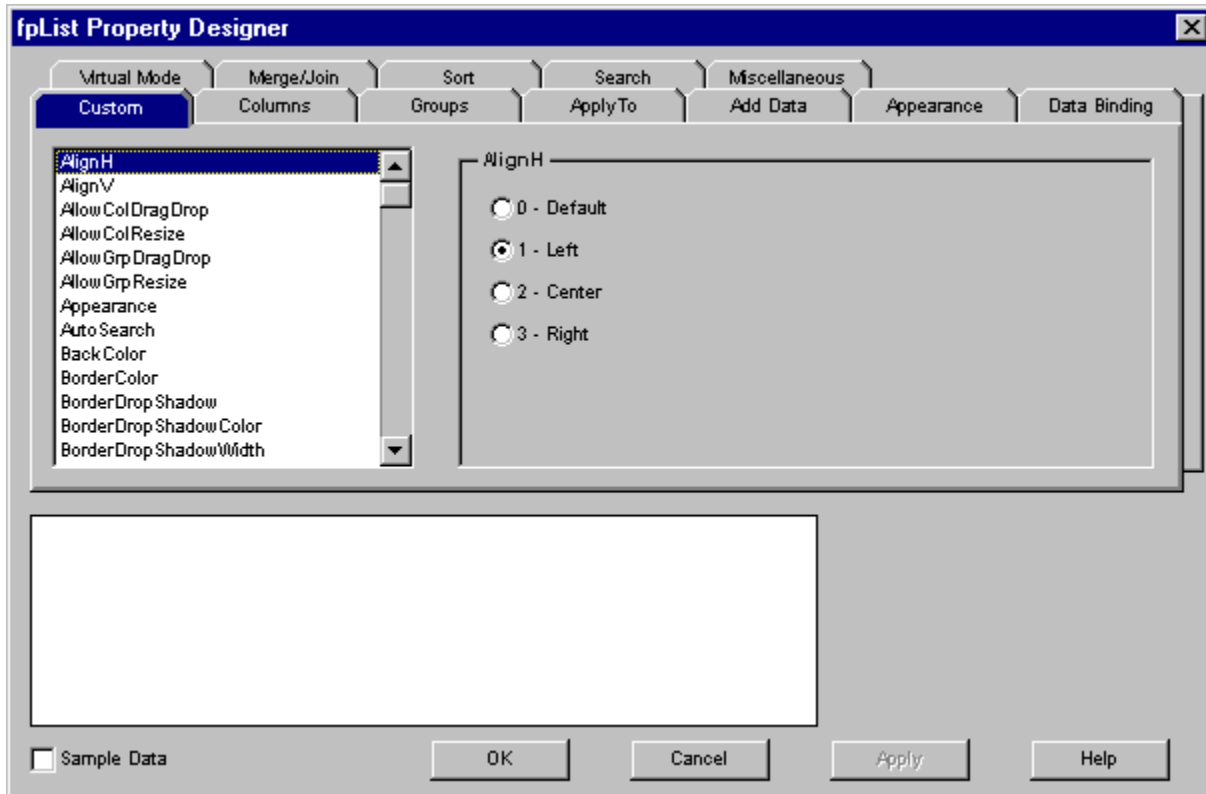
The List Pro fpCombo and fpList controls are available as 16- and 32-bit DLL controls for C and C++ users. For detailed information about DLL controls and the List Pro DLL controls, see [Using DLL Controls](#).

ActiveX Controls

Our List Pro fpCombo and fpList ActiveX controls offer the benefits of a new generation of controls. For detailed information about ActiveX controls and the List Pro ActiveX controls, see [Using ActiveX Controls](#).

FarPoint Property Designer

The FarPoint Property Designer provides a quick and simple way for you to design columns in a List Pro ActiveX or VBX control. The FarPoint Property Designer pages provide access to most design-time and some run-time properties. The following figure shows the Property Designer pages for the fpList control.



For general information on the FarPoint Property Designer, see [Using the FarPoint Property Designer](#). Instructions for using the FarPoint Property Designer to create and customize your List Pro control are provided in the [How-to-Guides](#).

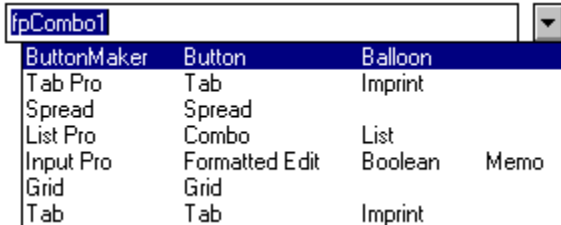
List Pro Tutorial

If you are unfamiliar with the FarPoint combo box and list box controls or if you want to see how some of the new List Pro features work, try the [tutorial](#). The tutorial guides you through creation of a sample project using code or the [FarPoint Property Designer](#).

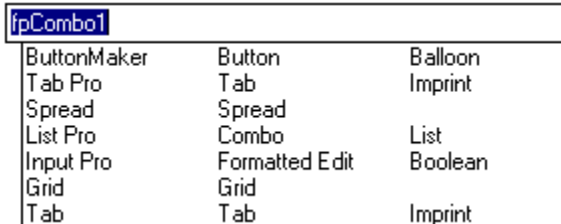
Combo Box Styles

An fpCombo control can be set to one of three styles: drop-down combo, simple combo, or drop-down list.

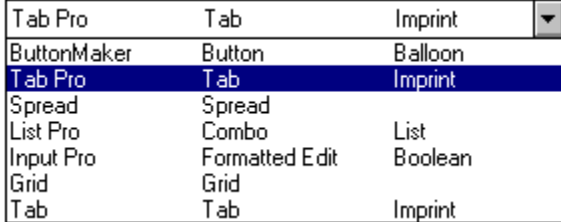
The drop-down combo style includes a drop-down list and an edit field, letting the user select from the list or type in the edit field.



The simple combo style includes an edit field and a list that is always displayed. The user can select from the list or type in the edit field.



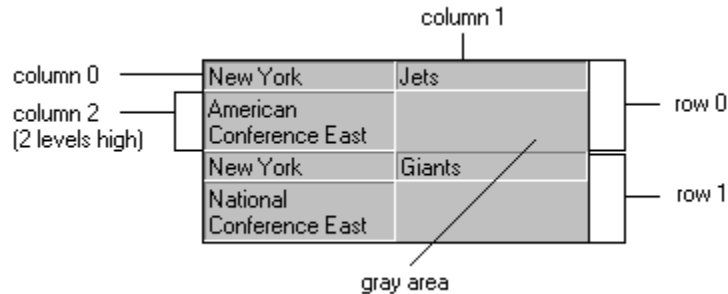
The drop-down list style contains a drop-down list and a static field, allowing the user to select only from the drop-down list.



For more information on choosing the fpCombo control style, see [Choosing the fpCombo Control Style](#).

Columns, Rows, and Cells

The fpCombo and fpList controls can display data in single- or multiple-column lists. Traditional combo and list boxes display columns from left to right across the control, creating a row. With List Pro controls, you can stack columns in a row, thereby creating multiple levels. If you have multiple levels of columns, for each column in a row, you can specify the level number and the height in levels. The following list box has three columns in each row. Each row has three levels. Columns 0 and 1 are on the first level. Column 2 is on the second level and is two levels high.



For more information on creating multiple levels, see [Creating Levels of Columns Within a Row](#).

You can duplicate or clone a column in a List Pro control. All attributes of a specified column except for the data, identification number, name, and data field setting (for bound columns) are duplicated in the cloned column. The cloned column is inserted to the right of the specified column. For more information, see [Cloning Columns](#).

A List Pro combo box or list box can be thought of as a tabular grid of cells. A cell is the intersection of a column and a row. You refer to cells using the column reference and row number. The following example shows 12 cells in the list box.

Name	Date	Cost (\$)
Hazel	1954	457M
Hugo	1989	7M
Bertha	1996	2.1B
Fran	1996	8B

Col = 2, Row = 0

Col = 0, Row = 3 Col = 1, Row = 2

You can merge adjacent cells in a column or row that contain the same text. You can merge columns, rows, or both. For example, assume you have the following list box.

Product	Week Ending	Units Sold
ButtonMaker	37	42
ButtonMaker	38	47
Spread 2.5	37	235
Spread 2.5	38	101
Input Pro	37	177
Input Pro	38	322

If you merge the first column, the control appears like this:

Product	Week Ending	Units Sold
ButtonMaker	37	42
	38	47
Spread 2.5	37	235
	38	101
Input Pro	37	177
	38	322

For more information, see [Merging Columns or Rows](#).

You can change the row height and column width to accommodate both text and graphics. For more information, see [Customizing Columns](#).

You can move columns by using the drag-drop method, changing the column level, or defining the position of a column within the control. You can resize columns by dragging the column or header boundary. For more information, see [Moving Columns in the Control](#) and [Resizing Columns](#). You can lock columns to prevent resizing and you can freeze columns to prevent moving using the drag-drop method. For more information on locking and freezing columns, see [Customizing Columns](#).

You can reference columns by column index number, column name, or column identifier number. The column index number is based on the physical position of the column in a row. The column name and identifier number are unique. For more information, see [Referencing a Column](#).

Groups

Columns in multiple-column fpCombo or fpList controls can be grouped together. These columns are children of the group. Groups can have groups as children or columns as children, but not a combination of the two. A column can be a child of a group that is a child of another group.

In the following example, the "Parents" group is a child of the parent group "Airedales." The "Sire" and "Dam" columns are children of the parent group "Parents."

Airedales			
Parents		Birth Information	
Sire	Dam	DOB	Number in Litter
Wanobi	Caddy	12/12/93	12
Lukas	Kamala	9/8/95	8
Bark Vader	BJ	1/28/87	8
Count	Tasha	3/18/91	5
Chewie	Dixie	7/3/92	7

You can reference groups by group index number, group name, or group identifier number. The group index number is based on the physical position of the group in a row. The group name and identifier number are unique. For more information, see [Referencing a Group](#).

For more information on groups, see [Working with Groups](#).

Headers

When you have a multiple-column fpCombo or fpList control, it is useful to display a header for each column. If you create groups for the control, you can also display group headers. For example, in the following figure "Parents" and "Birth Information" are group headers. You could specify "Sire", "Dam", "DOB", and "Number in Litter" as group headers or column headers, but only columns can contain data.

Parents		Birth Information	
Sire	Dam	DOB	Number in Litter
Chewie	Dixie	7/3/92	7
Count	Tasha	3/18/91	5
Bark Vader	BJ	1/28/87	8
Lukas	Kamala	9/8/95	8
Wanobi	Caddy	12/12/93	12

By default, the column header of a bound fpCombo or fpList control displays the name of the associated database field. However, you can create user-defined column headers for bound fpCombo and fpList controls.

For information on creating column and group headers, see [Providing Column Headers](#) and [Providing Group Headers](#).

Text and Graphics

You can display graphics, text, or both as list items in a single- or multiple-column fpCombo or fpList control. You can horizontally and vertically align list contents. If the list item contains both text and graphics, you can align them in relation to one another. You can also display text horizontally or vertically and you can rotate text and graphics 90, 180, or 270 degrees from the horizontal position. For more information, see [Aligning Text and Graphics](#) and [Orienting Text and Graphics](#).

You can customize the font and font characteristics of both the list text and the header text. For more information, see [Changing Text Color and Fonts](#).

You can give list text or header text a three-dimensional appearance by specifying the display, position, and color of the highlight and shadow text. For more information, see [Creating Three-Dimensional Text](#).

Appearance

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Border Appearance](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Lines](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Drop Shadow](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

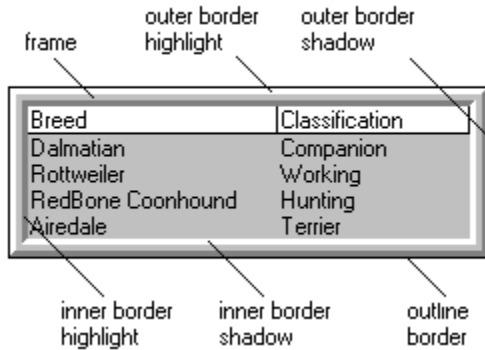
[Colors](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

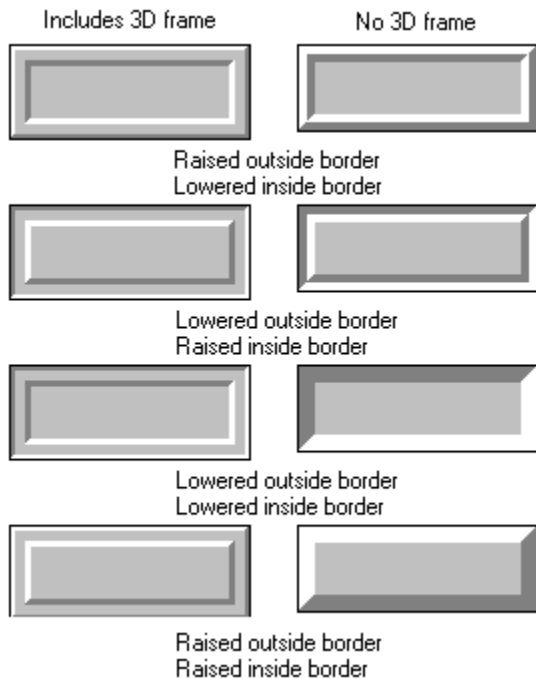
[Scroll Bars](#)

Border Appearance

The fpCombo and fpList controls can display a three-dimensional frame and borders. The inner and outer three-dimensional borders are made up of highlight and shadow portions. The frame represents the space between the inner and outer borders. In addition, the control can display an outline border and you can specify its color, width, and style. An fpList control with three-dimensional borders, a frame, and an outline border is shown in the following figure.



The following examples show some of the border combinations that can be created using the border properties.



You can use one of three predefined appearance styles (flat, three-dimensional, and three-dimensional with a border) or you can customize the borders. For more information on creating a three-dimensional appearance, see [Using Predefined Appearance Styles](#). For information on customizing the controls borders, see [Specifying the Border Appearance](#) and [Creating a Frame](#).

Lines

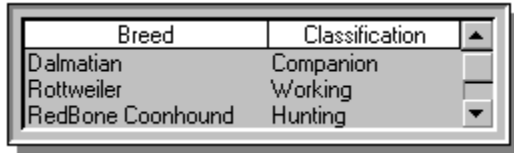
Between list items you can display horizontal lines, vertical lines, or both. You can display flat lines, three-dimensional lines, or both. The following example shows three-dimensional lines with a lowered appearance and horizontal and vertical flat lines.

Breed	Classification
Dalmatian	Companion
Rottweiler	Working
RedBone Coonhound	Sporting
Airedale	Terrier

For more information, see [Working with Lines](#).

Drop Shadow

You can display a drop shadow for the fpCombo or fpList control, as shown in the following figure. You can specify that the drop shadow is always displayed or you can specify that it is displayed only when the control has the focus. You can customize the color and width of the drop shadow.



Breed	Classification
Dalmatian	Companion
Rottweiler	Working
RedBone Coonhound	Hunting

For more information, see [Displaying Drop Shadows](#).

Colors

You can customize the colors of many different areas of the fpCombo and fpList controls.

To change the color of the . . .

Outline border
Three-dimensional inner and outer borders
Frame
Drop shadow
Border gray area
Background
Lines
Text
Three-dimensional text
List gray area

See . . .

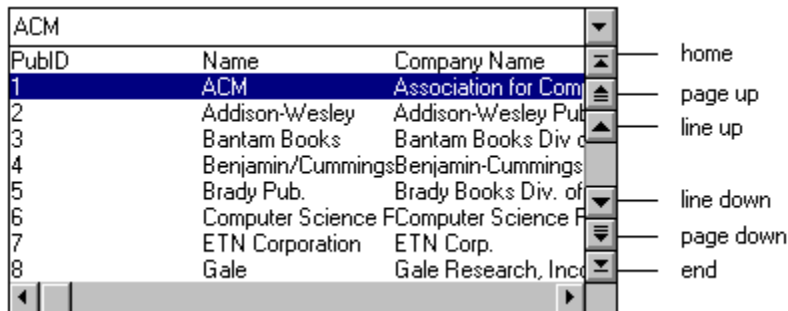
[Specifying the Border Appearance](#)
[Specifying the Border Appearance](#)
[Creating a Frame](#)
[Displaying Drop Shadows](#)
[Displaying Drop Shadows](#)
[Changing the Background Color](#)
[Customizing Lines](#)
[Changing Text Color and Fonts](#)
[Creating Three-Dimensional Text](#)
[Changing the List Gray Area Color](#)

Scroll Bars

You can control the display of both the horizontal and vertical scroll bars. For the horizontal scroll bar, you can specify the increment by which the list scrolls.

You can create a customized vertical scroll bar that lets you browse through the database when the fpCombo or fpList control is in virtual mode. You can use a conventional scroll bar while using virtual mode, but the scroll box will not accurately reflect the current position in the database unless you provide the number of records in the database.

The following example shows a drop-down fpCombo control that is bound to a database and displays a customized vertical scroll bar.



For more information on conventional scroll bars, see [Displaying and Customizing Scroll Bars](#). For more information on the customized vertical scroll bar, see [Using Virtual Mode](#).

List Items

You can use the fpCombo and fpList controls to display data lists either that you supply or that are read from a database.

By default, the number of items displayed in an fpCombo or fpList control is limited by the size of the control itself. However, for a drop-down list box, you can specify the number of items displayed. For more information, see [Specifying the Number of Rows Displayed in the Drop-Down List](#).

For a single-column fpCombo or fpList control, you can specify that the single column is displayed wrapped into side-by-side columns. For more information, see [Wrapping Text in a List Pro Control](#).

You can add items to a control list by using the List Pro InsertRow property, by binding the control to a database, or by using the Visual Basic AddItem method. You can remove items from a control list by using the List Pro Action property or the Visual Basic RemoveItem method. For more information, see [Adding List Items](#) and [Removing List Items](#).

At run time, you can select a list item, return the index of a list item, or return the value of a specific or selected list item. For more information, see [Accessing List Items](#).

In an fpCombo or fpList control, you can search for a specific list item. You can search for list items by typing characters or by supplying a search string. The search method for an fpCombo control depends on the control style. If the fpCombo or fpList control has multiple columns, you can specify which column to search. For more information, see [Searching for List Items](#).

You can sort fpCombo or fpList control list items to arrange them in alphabetical or numeric order. You can sort list items in ascending or descending order. By default, when you add an item to a list, the list is resorted and the item is inserted in the correct order. You can temporarily suspend sorting when you are adding or removing large numbers of items to help speed the process. If you have a multiple-column fpCombo or fpList control, you can specify the sequence in which the columns are sorted. You can also specify the data type of a column to make the results more accurate. For more information, see [Sorting List Items](#).

When items are selected in a list, they can have a different appearance from the non-selected items. You can also specify that a different picture be displayed when an item is selected. You can specify that the selected items display a three-dimensional appearance. If three-dimensional lines are displayed, you can change the width and highlight and shadow colors of the lines. You can also customize the background and text colors of the selected items, and you can specify that a focus rectangle be drawn around them. For more information, see [Customizing the Appearance of Selected Items](#).

Data Binding

List Pro controls work with the Visual Basic Data control and support any data access features provided by the Data control. The fpCombo and fpList controls can be bound to any type of database field except binary fields.

Working through Visual Basics Data control, List Pro controls provide access to more than one database. You can bind the edit field and the list in an fpCombo control to different Data controls and to different data fields. You can specify which data from the database bound to the list is displayed in the edit field. You can also specify which data from the database bound to the list is written to the database field bound to the edit field. In a multiple-column fpCombo or fpList control, you can bind each column to a separate database field. For more information, see [Binding a Control to a Database](#) and [Binding Columns to Fields in a Database](#).

The fpCombo and fpList controls can be bound to a Data control on another form. For more information, see [Binding to a Data Control on a Different Form](#).

The fpCombo and fpList controls display values exactly as they are stored in the database. However, you can apply a format string to list items or to a specific column to format the data.

The list in an fpCombo or fpList control is read-only and is designed for viewing database values. However, if the user changes the current record through another control, the fpCombo or fpList control reflects the change when the Data control is refreshed.

The selected item in a bound fpCombo or fpList control is the value in the current record. Selecting another item automatically moves the Data control to another record. Likewise, by default, clicking the Data control to change the current record changes the item selected in the fpCombo or fpList control. You can customize the synchronization between the fpCombo or fpList control and the Data control. For more information, see [Working with Databases](#).

Virtual Mode

Virtual mode lets the fpCombo or fpList control read in only the amount of data it needs to fill the displayed list. Therefore, virtual mode increases responsiveness and conserves system resources.

Without using virtual mode, the fpCombo or fpList control can take a long time to display data when the number of list items is very large or when the control is bound to a very large database table. The fpCombo or fpList control must read all supplied data or the entire database into memory before displaying the list. In virtual mode, the fpCombo or fpList control reads ahead only when the user wants to scroll through the list to display additional values.

For more information on using virtual mode, see [Using Virtual Mode](#).

List Pro Components

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Components of the fpCombo DLL Control](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Components of the fpCombo ActiveX and VBX Controls](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Components of the fpList DLL Control](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Components of the fpList ActiveX and VBX Controls](#)

Components of the fpCombo DLL Control

Functions let you set characteristics of the fpCombo control. Throughout the help, discussions of setting characteristics have been combined: property descriptions are intended to also describe the corresponding DLL functions. Therefore, if you use the DLL control, refer to the corresponding property topic in the property reference in the online Reference Guide for information about performing your task using functions.

Messages convey information from the fpCombo control to your application. Styles let you customize the fpCombo control.

The following topics provide complete lists of the functions, messages, and styles available for the fpCombo control.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Functions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Messages](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Styles](#)

fpCombo Control Functions

The following list includes all the functions of the fpCombo control. For a complete description of each function, refer to the corresponding property topic in the online [Reference Guide](#).

GetAbout	GetVRowCount
GetAlignH	GetVScrollSpecial
GetAlignV	GetVScrollSpecialType
GetAllowColDragDrop	GetWidth
GetAllowColResize	GetWrapList
GetAllowGrpDragDrop	GetWrapWidth
GetAllowGrpResize	SetAction
GetAppearance	SetAlignH
GetApplyTo	SetAlignV
GetAutoSearch	SetAllowColDragDrop
GetBackColor	SetAllowColResize
GetBorderColor	SetAllowGrpDragDrop
GetBorderDropShadow	SetAllowGrpResize
GetBorderDropShadowColor	SetAppearance
GetBorderDropShadowWidth	SetApplyTo
GetBorderGrayAreaColor	SetAutoSearch
GetBorderStyle	SetBackColor
GetBorderWidth	SetBorderColor
GetCol	SetBorderDropShadow
GetColFromID	SetBorderDropShadowColor
GetColFromName	SetBorderDropShadowWidth
GetColHeaderText	SetBorderGrayAreaColor
GetColHide	SetBorderStyle
GetColID	SetBorderWidth
GetColLevel	SetCol
GetColLevelHeight	SetColFromID
GetColList	SetColFromName
GetColLockResize	SetColHeaderText
GetColMerge	SetColHide
GetColName	SetColID
GetColParentGroup	SetColLevel
GetColPos	SetColLevelHeight
GetColPosInParent	SetColList
GetColsFrozen	SetColLockResize
GetColSortDataType	SetColMerge
GetColSorted	SetColName
GetColSortSeq	SetColParentGroup
GetColText	SetColPos
GetColumnEdit	SetColPosInParent
GetColumnHeaderHeight	SetColsFrozen
GetColumnHeaderShow	SetColSortDataType
GetColumnLevels	SetColSorted
GetColumns	SetColText
GetColumnSearch	SetColumnEdit
GetColumnSeparatorChar	SetColumnHeaderHeight
GetColumnWidthScale	SetColumnHeaderShow
GetColWidth	SetColumnLevels
GetComboGap	SetColumns
GetDragIcon	SetColumnSearch
GetDragMode	SetColumnSeparatorChar
GetEditHeight	SetColumnWidthScale
GetEnabled	SetColWidth
GetEnableKeyEvents	SetComboGap
GetEnableMouseEvents	SetDragIcon
GetEnableTopChangeEvent	SetDragMode
GetExtendCol	SetEditHeight

GetExtendRow	SetEnabled
GetFontBold	SetEnableKeyEvents
GetFontEmpty	SetEnableMouseEvents
GetFontItalic	SetEnableTopChangeEvent
GetFontName	SetExtendCol
GetFontSize	SetExtendRow
GetFontStrikethru	SetFontBold
GetFontUnderline	SetFontEmpty
GetForeColor	SetFontItalic
GetGrayAreaColor	SetFontName
GetGroupHeaderHeight	SetFontSize
GetGroupHeaderShow	SetFontStrikethru
GetGroups	SetFontUnderline
GetGrp	SetGrayAreaColor
GetGrpFromID	SetGroupHeaderHeight
GetGrpFromName	SetGroupHeaderShow
GetGrpHeaderText	SetGroups
GetGrpHide	SetGrp
GetGrpID	SetGrpFromID
GetGrpLockResize	SetGrpFromName
GetGrpName	SetGrpHeaderText
GetGrpParentGroup	SetGrpHide
GetGrpPos	SetGrpID
GetGrpPosInParent	SetGrpLockResize
GetGrpsFrozen	SetGrpName
GetGrpWidth	SetGrpParentGroup
GetHeight	SetGrpPos
GetHelpContextID	SetGrpPosInParent
GetHighestPrecedence	SetGrpsFrozen
GetHwnd	SetGrpWidth
GetIndex	SetHeight
GetItemData	SetHelpContextID
GetJoinID	SetHighestPrecedence
GetLeft	SetInsertRow
GetLine3DDark	SetItemData
GetLine3DLight	SetJoinID
GetLine3DWidth	SetLeft
GetLineApplyTo	SetLine3DDark
GetLineColor	SetLine3DLight
GetLineStyle	SetLine3DWidth
GetLineWidth	SetLineApplyTo
GetList	SetLineColor
GetList3DText	SetLineStyle
GetList3DTextHighlightColor	SetLineWidth
GetList3DTextOffset	SetList
GetList3DTextShadowColor	SetList3DText
GetListApplyTo	SetList3DTextHighlightColor
GetListCount	SetList3DTextOffset
GetListDown	SetList3DTextShadowColor
GetListGrayAreaColor	SetListApplyTo
GetListIndex	SetListCount
GetListLeftOffset	SetListDown
GetListWidth	SetListGrayAreaColor
GetMaxDrop	SetListIndex
GetMaxEditLen	SetListLeftOffset
GetMergeAdjustView	SetListWidth
GetMouseOverArea	SetMaxDrop
GetMouseOverCol	SetMaxEditLen
GetMouseOverColHeader	SetMergeAdjustView
GetMouseOverGrp	SetMousePointer

GetMouseOverGrpHeader	SetMultiLine
GetMouseOverRow	SetName
GetMousePointer	SetNextSel
GetMultiLine	SetNoIntegralHeight
GetName	SetPicture
GetNewIndex	SetPictureAlignH
GetNextSel	SetPictureAlignV
GetNoIntegralHeight	SetPictureSel
GetParent	SetReadOnly
GetPicture	SetRow
GetPictureAlignH	SetRowHeight
GetPictureAlignV	SetRowMerge
GetPictureSel	SetScrollBarH
GetReadOnly	SetScrollBarV
GetRow	SetScrollHInc
GetRowHeight	SetScrollHScale
GetRowMerge	SetSearchIgnoreCase
GetScrollBarH	SetSearchIndex
GetScrollBarV	SetSearchMethod
GetScrollHInc	SetSearchText
GetScrollHScale	SetSelDrawFocusRect
GetSearchIgnoreCase	SetSelLength
GetSearchIndex	SetSelStart
GetSearchMethod	SetSelText
GetSearchText	SetSorted
GetSelDrawFocusRect	SetSortState
GetSelLength	SetStyle
GetSelStart	SetTabIndex
GetSelText	SetTabStop
GetSorted	SetTag
GetSortState	SetText
GetStyle	SetTextOrientation
GetTabIndex	SetThreeDFrameColor
GetTabStop	SetThreeDFrameWidth
GetTag	SetThreeDInsideHighlightColor
GetText	SetThreeDInsideShadowColor
GetTextOrientation	SetThreeDInsideStyle
GetThreeDFrameColor	SetThreeDInsideWidth
GetThreeDFrameWidth	SetThreeDOnFocusInvert
GetThreeDInsideHighlightColor	SetThreeDOutsideHighlightColor
GetThreeDInsideShadowColor	SetThreeDOutsideShadowColor
GetThreeDInsideStyle	SetThreeDOutsideStyle
GetThreeDInsideWidth	SetThreeDOutsideWidth
GetThreeDOnFocusInvert	SetTop
GetThreeDOutsideHighlightColor	SetTopIndex
GetThreeDOutsideShadowColor	SetVirtualMode
GetThreeDOutsideStyle	SetVirtualPagesAhead
GetThreeDOutsideWidth	SetVirtualPageSize
GetTop	SetVisible
GetTopIndex	SetVRowCount
GetVirtualMode	SetVScrollSpecial
GetVirtualPagesAhead	SetVScrollSpecialType
GetVirtualPageSize	SetWidth
GetVisible	SetWrapList
GetVisibleRows	SetWrapWidth

Components of the fpCombo ActiveX and VBX Controls

You can use the FarPoint Property Designer pages, the standard property browser, or code to customize the control. The FarPoint Property Designer pages provide a familiar notebook interface that organizes properties by task. For example, all the

properties for virtual mode are available on one page. For descriptions of each designer page, see [FarPoint Property Designer Pages](#).

If you are using code, properties and events let you customize the control for your interface.

The following topics provide complete lists of the properties, events, and functions available for the fpCombo control. Note that if you are using the ActiveX control, you can access some of these properties on the FarPoint Property Designer pages, but others may not be available or may be available only through the property browser, depending on your development environment.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Properties](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Events](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Functions and Methods](#)

Components of the fpList DLL Control

Functions let you set characteristics of the fpList control. Throughout the help, discussions of setting characteristics have been combined: property descriptions are intended to also describe the corresponding DLL functions. Therefore, if you use the DLL control, refer to the corresponding property topic in the property reference in the online [Reference Guide](#) for information about performing your task using functions.

Messages convey information from the fpList control to your application. Styles let you customize the fpList control.

The following topics provide complete lists of the functions, messages, and styles available for the fpList control.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Functions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Messages](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Structures](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Styles](#)

fpList Control Functions

The following list includes all the functions of the fpList control. For a complete description of each function, refer to the corresponding property topic in the online [Reference Guide](#).

GetAbout	GetVScrollSpecial
GetAlignH	GetVScrollSpecialType
GetAlignV	GetWidth
GetAllowColDragDrop	GetWrapList
GetAllowColResize	GetWrapWidth
GetAllowGrpDragDrop	SetAction
GetAllowGrpResize	SetAlignH
GetAppearance	SetAlignV
GetAutoSearch	SetAllowColDragDrop
GetBackColor	SetAllowColResize
GetBorderColor	SetAllowGrpDragDrop
GetBorderDropShadow	SetAllowGrpResize
GetBorderDropShadowColor	SetAppearance
GetBorderDropShadowWidth	SetAutoSearch
GetBorderGrayAreaColor	SetBackColor
GetBorderStyle	SetBorderColor
GetBorderWidth	SetBorderDropShadow
GetCol	SetBorderDropShadowColor
GetColFromID	SetBorderDropShadowWidth
GetColFromName	SetBorderGrayAreaColor
GetColHeaderText	SetBorderStyle
GetColHide	SetBorderWidth
GetColID	SetCol
GetColLevel	SetColFromID
GetColLevelHeight	SetColFromName
GetColList	SetColHeaderText
GetColLockResize	SetColHide
GetColMerge	SetColID
GetColName	SetColLevel
GetColParentGroup	SetColLevelHeight
GetColPos	SetColList
GetColPosInParent	SetColLockResize
GetColsFrozen	SetColMerge
GetColSortDataType	SetColName
GetColSorted	SetColParentGroup
GetColSortSeq	SetColPos
GetColText	SetColPosInParent
GetColumnHeaderHeight	SetColsFrozen
GetColumnHeaderShow	SetColSortDataType
GetColumnLevels	SetColSorted
GetColumns	SetColSortSeq
GetColumnSearch	SetColText
GetColumnSeparatorChar	SetColumnHeaderHeight
GetColumnWidthScale	SetColumnHeaderShow
GetColWidth	SetColumnLevels
GetDragIcon	SetColumns
GetDragMode	SetColumnSearch
GetEnabled	SetColumnSeparatorChar
GetEnableKeyEvents	SetColumnWidthScale
GetEnableMouseEvents	SetColWidth
GetEnableTopChangeEvent	SetDragIcon
GetExtendCol	SetDragMode
GetExtendRow	SetEnabled
GetFontBold	SetEnableKeyEvents
GetFontEmpty	SetEnableMouseEvents
GetFontItalic	SetEnableTopChangeEvent

GetFontName	SetExtendCol
GetFontSize	SetExtendRow
GetFontStrikethru	SetFontBold
GetFontUnderline	SetFontEmpty
GetForeColor	SetFontItalic
GetGroupHeaderHeight	SetFontName
GetGroupHeaderShow	SetFontSize
GetGroups	SetFontStrikethru
GetGrp	SetFontUnderline
GetGrpFromID	SetForeColor
GetGrpFromName	SetGroupHeaderHeight
GetGrpHeaderText	SetGroupHeaderShow
GetGrpHide	SetGroups
GetGrpID	SetGrp
GetGrpLockResize	SetGrpFromID
GetGrpName	SetGrpFromName
GetGrpParentGroup	SetGrpHeaderText
GetGrpPos	SetGrpHide
GetGrpPosInParent	SetGrpLockResize
GetGrpsFrozen	SetGrpName
GetGrpWidth	SetGrpParentGroup
GetHeight	SetGrpPos
GetHelpContextID	SetGrpPosInParent
GetHighestPrecedence	SetGrpsFrozen
GetHwnd	SetGrpWidth
GetIndex	SetHeight
GetItemData	SetHelpContextID
GetJoinID	SetHighestPrecedence
GetLeft	SetInsertRow
GetLine3DDark	SetItemData
GetLine3DLight	SetJoinID
GetLine3DWidth	SetLeft
GetLineApplyTo	SetLine3DDark
GetLineColor	SetLine3DLight
GetLineStyle	SetLine3DWidth
GetLineWidth	SetLineApplyTo
GetList	SetLineColor
GetList3DText	SetLineStyle
GetList3DTextHighlightColor	SetLineWidth
GetList3DTextOffset	SetList
GetList3DTextShadowColor	SetList3DText
GetListApplyTo	SetList3DTextHighlightColor
GetListCount	SetList3DTextOffset
GetListGrayAreaColor	SetList3DTextShadowColor
GetListIndex	SetListApplyTo
GetMergeAdjustView	SetListCount
GetMouseOverCol	SetListGrayAreaColor
GetMouseOverColHeader	SetListIndex
GetMouseOverGrp	SetMergeAdjustView
GetMouseOverGrpHeader	SetMousePointer
GetMouseOverRow	SetMultiLine
GetMousePointer	SetMultiSelect
GetMultiLine	SetName
GetMultiSelect	SetNextSel
GetName	SetNoIntegralHeight
GetNewIndex	SetPicture
GetNextSel	SetPictureAlignH
GetNoIntegralHeight	SetPictureAlignV
GetParent	SetPictureSel
GetPicture	SetReadOnly

GetPictureAlignH	SetRow
GetPictureAlignV	SetRowHeight
GetPictureSel	SetRowMerge
GetReadOnly	SetScrollBarH
GetRow	SetScrollBarV
GetRowHeight	SetScrollHInc
GetRowMerge	SetScrollHScale
GetScrollBarH	SetSearchIgnoreCase
GetScrollBarV	SetSearchIndex
GetScrollHInc	SetSearchMethod
GetScrollHScale	SetSearchText
GetSearchIgnoreCase	SetSelDrawFocusRect
GetSearchIndex	SetSelected
GetSearchMethod	SetSelMax
GetSearchText	SetSorted
GetSelCount	SetSortState
GetSelDrawFocusRect	SetTabIndex
GetSelected	SetTabStop
GetSelMax	SetTag
GetSorted	SetText
GetSortState	SetTextOrientation
GetTabIndex	SetThreeDFrameColor
GetTabStop	SetThreeDFrameWidth
GetTag	SetThreeDInsideHighlightColor
GetText	SetThreeDInsideShadowColor
GetTextOrientation	SetThreeDInsideStyle
GetThreeDFrameColor	SetThreeDInsideWidth
GetThreeDFrameWidth	SetThreeDOnFocusInvert
GetThreeDInsideHighlightColor	SetThreeDOutsideHighlightColor
GetThreeDInsideShadowColor	SetThreeDOutsideShadowColor
GetThreeDInsideStyle	SetThreeDOutsideStyle
GetThreeDInsideWidth	SetThreeDOutsideWidth
GetThreeDOnFocusInvert	SetTop
GetThreeDOutsideHighlightColor	SetTopIndex
GetThreeDOutsideShadowColor	SetVirtualMode
GetThreeDOutsideStyle	SetVirtualPagesAhead
GetThreeDOutsideWidth	SetVirtualPageSize
GetTop	SetVisible
GetTopIndex	SetVRowCount
GetVirtualMode	SetVScrollSpecial
GetVirtualPagesAhead	SetVScrollSpecialType
GetVirtualPageSize	SetWidth
GetVisible	SetWrapList
GetVisibleRows	SetWrapWidth
GetVRowCount	

Components of the fpList ActiveX and VBX Controls

You can use the FarPoint Property Designer pages, the standard property browser, or code to customize the control. The FarPoint Property Designer pages provide a familiar notebook interface that organizes properties by task. For example, all the properties for virtual mode are available on one page. For descriptions of each designer page, see [FarPoint Property Designer Pages](#).

If you are using code, properties and events let you customize the control for your interface.

The following topics provide complete lists of the properties, events, and functions available for the fpList control. Note that if you are using the ActiveX control, you can access some of these properties on the FarPoint Property Designer pages, but others may not be available or may be available only through the property browser, depending on your development environment.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Properties](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Events](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Functions and Methods](#)

List Pro Tutorial

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Tutorial Example Project](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Step 1: Plan the Control Layout](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Step 2: Create the Data Control](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Step 3: Create the List Box and Bind It to the Database](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Step 4: Bind Columns in the Control](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Step 5: Customize the Control](#)

Tutorial Example Project

Note Before starting this tutorial, you should be familiar with the List Pro control features described in [List Pro Features](#), and with the FarPoint Property Designer features described in [Using the FarPoint Property Designer](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Project Assumptions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Format](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Database Information](#)

Using this tutorial, you will design a relatively simple list box. This tutorial does not illustrate every List Pro property, but it does illustrate new features such as groups, parent-child relationships, multiple levels, appearance, and applying properties to specific portions of the control.

Starting with [Step 1](#), the tutorial provides step-by-step instructions to:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Plan the controls layout

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Build the database front end

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Create the list box and link it to the database

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Bind columns in the control to specific fields in the database using code or using the FarPoint

Property Designer

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Customize the control using code or using the FarPoint Property Designer

PD	RD	WR
✓	✓	✓

In the tutorial, this symbol indicates a cross-reference to topics in the [How-to Guides](#). Read these topics to find out more information about the task you are performing.

Project Assumptions

Assume a local kennel club wants you to create an application that displays information from a database containing the following data fields:

- Owner Name
- Address
- Phone
- Breed
- Dog Name
- Color
- Sire
- Dam

Format

For this project, categorize the data into three groups: owner information, registrant information, and lineage. The owner information group includes the owner name, complete address, and phone number. The registrant information group includes the breed, dog name, and color. The lineage group includes the sire and dam information.

You will create a list box that displays the information in the format shown in the following figure. Each row contains eight columns, and each row corresponds to a record in the database. Each row has three levels.

group headers		Owner Info		Registrant		Lineage		
column headers		Name	Phone	Breed	Name	Sire		level0
		Address			Color	Dam		level1
								level2
(data rows will display here)								

You will make the columns children of the three groups and position these columns within the groups.

Database Information

If you are using the ActiveX or VBX control, you can bind the fpList control to a database (TUTOR.MDB) that contains the kennel records. If you are using the DLL control, you can create and design the list box, but you cannot bind it to a database.

Note TUTOR.MDB is included on the List Pro installation disks. If you did a complete default installation of List Pro, the database is in your LSTPRO20\SAMPLES\VBX\VB\TUTOR directory.

TUTOR.MDB contains the table KENNEL CLUB. The table contains ten records and each record has eight fields, as listed in the following table:

Field number	Field name
1	Owner Name
2	Address
3	Phone
4	Breed
5	Dog Name
6	Color
7	Sire
8	Dam

Step 1: Plan the Control Layout

You will probably find it helpful to sketch a preliminary layout of your groups and columns. These sketches give you an idea of the order in which you should assign columns to groups and position columns within those groups. Remember that columns and groups are numbered from left to right and top to bottom, within their parent group, if any. With that in mind, the following figures explain how to plan your list box based on the finished format shown in [Format](#).

Look at the Owner Info group.

Owner Info (Grp = 0)	
Name (Col = 0) (Pos in Parent = 0)	Phone (Col = 1) (Pos in Parent = 1)
Address (Col = 2, Level = 1, Level Height = 2) (Pos in Parent = 2)	

The Address column starts on level 1 and is two levels high. Notice how the columns are numbered from left to right and top to bottom within the group.

Look at the Registrant and Lineage groups.

Registrant (Grp = 1)		Lineage (Grp = 2)
Breed (Col = 3) (Level Height = 2) (Pos in Parent = 0)	Name (Col = 4) (Pos in Parent = 1)	Sire (Col = 6) (Pos in Parent = 0)
	Color (Col = 5, Level = 1) (Pos in Parent = 2)	Dam (Col = 7) (Level = 1) (Pos in Parent = 1)

In the Registrant group, the Breed column starts on level 0 and is two levels high. Notice again that the columns are numbered from left to right and top to bottom within each group.

With this plan in mind, you are ready to create the Data control to which you will link your list box. If you are ready to continue, go to [Step 2: Create the Data Control](#).

Step 2: Create the Data Control

▶ [Browser Instructions](#)

Overview

Note If you are using the DLL control, skip to [Step 5: Customize the Control](#). You cannot bind to the sample database for this example, but you can follow the steps to design the list box.

In this part of the tutorial, you will create the Data control and open the sample database.

[Print](#)

[Copy](#)

[Close](#)

To create the Data control and open the sample database

Browser

1. Start Visual Basic.
2. Select the Data control icon on the toolbar.
3. Draw a Data control on the form.
4. Click the Data control to select it.
5. Set the Name property of the Data control to TUTORIAL.
6. Set the DatabaseName property to C:\LSTPRO20\SAMPLES\VBX\VB\TUTOR\TUTOR.MDB.
7. Set the RecordSource property to Kennel Club.

The next step is to create the list box control and bind it to the database. If you are ready to continue, go to [Step 3: Create the List Box and Bind It to the Database](#).

Step 3: Create the List Box and Bind It to the Database

▶ [Browser Instructions](#)

Overview

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If you are using the DLL control, create the control as described in [Step 1](#) in the browser instructions, then skip to [Step 5: Customize the Control](#). You cannot bind to the sample database for this example, but you can follow the steps to design the list box.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

You must have the List Pro ActiveX or VBX control installed and, in the case of the ActiveX control, registered on your system.

In this part of the tutorial you will create the list box and bind it to the sample database.

[Print](#)

[Copy](#)

[Close](#)

Browser

To create the control and bind it to the database

1. To create the list box,
 - a. If you are using Visual Basic 3.0,
 - i. Choose Add File from the File menu.
 - ii. In the Add File dialog box, choose the directory or set the path to the directory that contains the List Pro VBX control, select the .VBX file, and choose OK.



The List Pro icons (list box) and



(combo box) appear on the toolbar. The FLPVBX20.VBX file appears in the Project window.



- iii. Add the LP.BAS constants file following steps 1.a.i 1.a.ii selecting the LP.BAS file instead of the .VBX file.

If you chose the default installation, this file is located in LSTPRO20\INCLUDE\ directory. The LP.BAS file appears in the Project window.

- b. If you are using Visual Basic 4.0,
 - i. Choose Custom Controls from the Tools menu.
 - ii. Choose FarPoint List Pro Controls from the Custom Controls dialog box.
 - iii. Choose OK.



The List Pro icons (list box) and



(combo box) appear on the toolbar.

- iv. Add the LP.BAS constants file following step 1.a.iii.
If you chose the default installation, this file is located in LSTPRO20\INCLUDE\ directory.

- c. Select the fpList icon on the toolbar.
 - d. Draw an fpList control on your form.
Draw the control about 6 inches wide. You can resize it later to fit the data.

2. To bind the fpList control to the database,
 - a. Click the fpList control to select it.
 - b. Set the DataSource property to TUTORIAL.
3. Press F5 or click the Visual Basic run button.

The list box displays all eight columns of database data. You can use the list box scroll bars or the Data controls slide buttons to move through the data.

Owner Name	Address	
Julia James	5008 Monitor Dr., Peak NC, 27052	(919) 555-0079
Tim Wrenn	321 Webster Street, Queens NY, 10163	(201) 555-7345
Scott Sain	2842 Avent Road, Madison WI, 53705	(608) 555-2482
Denise Lanier	5804 Dutch Creek Drive, Atlanta GA, 30	(404) 555-0180
Bruce Gibson	127 Charter Court, Boston MA, 02111	(413) 555-8648

The next step is to bind the columns in the control. If you are ready to continue, go to [Step 4: Bind Columns in the Control](#)

Step 4: Bind Columns in the Control

- ▶ [FarPoint Property Designer Instructions](#)
- ▶ [Code Instructions](#)

Overvi

ew

To correctly display the required database fields, you must bind each column in the control to a specific field.

[Print](#)

[Copy](#)

[Close](#)

To bind columns to specific database fields

Code

1. Set the [Columns](#) property as follows to display the eight columns for this tutorial.

Property	Setting
Columns	8

PD	RD	WR
✓	✓	✓

[Creating Multiple Columns](#)

2. Bind columns to specific database fields by setting the [Col](#) and [ColDataField](#) properties as follows.

Property	Setting	What this means
Col ColDataField	0 "#1"	Bind the first column to field 1 (Owner Name) in the database
Col ColDataField	1 "#2"	Bind the second column to field 2 (Address) in the database
Col ColDataField	2 "#3"	Bind the third column to field 3 (Phone) in the database
Col ColDataField	3 "#4"	Bind the fourth column to field 4 (Breed) in the database
Col ColDataField	4 "#5"	Bind the fifth column to field 5 (Dog Name) in the database
Col ColDataField	5 "#6"	Bind the sixth column to field 6 (Color) in the database
Col ColDataField	6 "#7"	Bind the seventh column to field 7 (Sire) in the database
Col ColDataField	7 "#8"	Bind the eighth column to field 8 (Dam) in the database

PD	RD	WR
✓	✓	✓

[Binding Columns to Fields in a Database](#)

The next step is to customize the control. If you are ready to continue, go to [Step 5: Customize the Control](#).

[Print](#)

[Copy](#)

[Close](#)

To bind columns to specific database fields

Property Designer

1. Start the property designer.

For more information, see [Starting the FarPoint Property Designer](#).

2. Display the eight columns for this tutorial by setting the [Columns](#) property.

On the [Specific subtab of the Columns designer page](#),

- a. Select the Columns property from the properties list box.
- b. Type 8 in the box under Columns in the property value area.

PD	RD	WR
✓	✓	✓

[Creating Multiple Columns](#)

3. Bind columns to specific database fields by setting the [Col](#) and [ColDataField](#) properties.

On the Specific subtab of the Columns designer page,

- a. Select column 0 from the Col drop-down list box under Individual.
- b. Select the ColDataField property from the properties list box.
- c. Choose Owner Name from the drop-down list box under ColDataField in the property value area.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

- d. Repeat step 3.a

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

7, binding the columns to the following data fields:

Column	Data Field
1	Address
2	Phone
3	Breed
4	Dog Name
5	Color
6	Sire
7	Dam

PD	RD	WR
✓	✓	✓

[Binding Columns to Fields in a Database](#)

The next step is to customize the control. If you are ready to continue, go to [Step 5: Customize the Control](#).

Step 5: Customize the Control

- ▶ [FarPoint Property Designer Instructions](#)
- ▶ [Code Instructions](#)

Overview

In this step you will create groups, display headers, and move the columns into groups. Because List Pro resizes columns when you move them, you will need to adjust their size. You will need to allow column and group resizing.

You will also customize the parts of the control as shown in the following table.

Part of the control	Customization										
Owner Info group	Width										
Registrant group	Width										
Lineage group	Width										
Address column	Display text in multiple lines										
Breed column	Display text in multiple lines										
Entire control	Light gray background color										
List items	Lowered three-dimensional appearance										
Column headers	<table border="1"> <tr> <td>PD</td> <td>RD</td> <td>WR</td> <td>RT</td> <td>DT</td> </tr> <tr> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> </tr> </table> Black background color	PD	RD	WR	RT	DT	✓	✓	✓	✓	✓
PD	RD	WR	RT	DT							
✓	✓	✓	✓	✓							
Group headers	<table border="1"> <tr> <td>PD</td> <td>RD</td> <td>WR</td> <td>RT</td> <td>DT</td> </tr> <tr> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> </tr> </table> White text color	PD	RD	WR	RT	DT	✓	✓	✓	✓	✓
PD	RD	WR	RT	DT							
✓	✓	✓	✓	✓							
Selected rows	Green background color <table border="1"> <tr> <td>PD</td> <td>RD</td> <td>WR</td> <td>RT</td> <td>DT</td> </tr> <tr> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> </tr> </table> Red background color	PD	RD	WR	RT	DT	✓	✓	✓	✓	✓
PD	RD	WR	RT	DT							
✓	✓	✓	✓	✓							
Line between rows	<table border="1"> <tr> <td>PD</td> <td>RD</td> <td>WR</td> <td>RT</td> <td>DT</td> </tr> <tr> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> </tr> </table> White text color Green	PD	RD	WR	RT	DT	✓	✓	✓	✓	✓
PD	RD	WR	RT	DT							
✓	✓	✓	✓	✓							

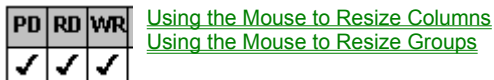
[Print](#) [Copy](#) [Close](#)

To customize the list box

Code

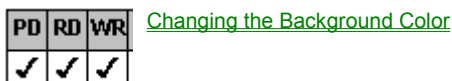
1. Allow resizing of the columns and groups by setting the [AllowColResize](#) and [AllowGrpResize](#) properties as follows.

Property	Setting (Constant)
AllowColResize	2 (LC_ALLOWCOLRESIZE_RESIZECOLORHEADER)
AllowGrpResize	2 (LC_ALLOWGRPRESIZE_RESIZEGRPORHEADER)



2. Change the background color of the list box by setting the [BackColor](#) property as follows.

Property	Setting
BackColor	RGB (192, 192, 192) 00C0C0C0



3. Create a lowered, three-dimensional appearance for lines by setting the [LineStyle](#) property as follows.

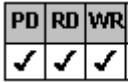
Property	Setting (Constant)
LineStyle	3 (LC_LINestyle_LOWERED)



[Customizing Lines](#)

4. Display the column headers and create three column levels by setting the [ColumnHeaderShow](#) and [ColumnLevels](#) properties as follows.

Property	Setting
ColumnHeaderShow	True
ColumnLevels	3



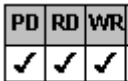
[Creating Multiple Columns](#)

[Creating Column Headers](#)

[Creating Levels of Columns Within a Row](#)

5. Create the three groups and display the group headers by setting the [Groups](#) and [GroupHeaderShow](#) properties as follows.

Property	Setting
Groups	3
GroupHeaderShow	True

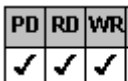


[Creating Groups](#)

[Creating Group Headers](#)

6. Specify pixels as the unit of measurement for the group widths by setting the [ColumnWidthScale](#) property as follows.

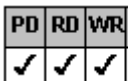
Property	Setting (Constant)
ColumnWidthScale	1 (LC_COLUMNWIDTHSCALE_PIXELS)



[Specifying the Group Width](#)

7. Define the name, width, and header text for each group by setting the [Grp](#), [GrpName](#), [GrpHeaderText](#), and [GrpWidth](#) properties as follows.

Property	Setting	What this means
Grp	0	Assign the name OI to the first group, specify the header text, set the width to 170 pixels
GrpName	"OI"	
GrpHeaderText	"Owner Info"	
GrpWidth	170	
Grp	1	Assign the name RG to the second group, specify the header text, set the width to 225 pixels
GrpName	"RG"	
GrpHeaderText	"Registrant"	
GrpWidth	225	
Grp	2	Assign the name LN to the third group, specify the header text, set the width to 125 pixels
GrpName	"LN"	
GrpHeaderText	"Lineage"	
GrpWidth	125	



[Referencing a Group](#)

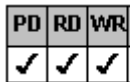
[Creating Group Headers](#)

[Specifying the Group Width](#)

8. Define the column names and custom header text by setting the [DataAutoHeadings](#), [DataAutoSizeCols](#), [Col](#), [ColName](#), and [ColHeaderText](#) properties as follows.

Property	Setting (Constant)	What this means
DataAutoHeadings	False	Use custom header text
DataAutoSizeCols	0 (LC_DATAAUTOSIZECOLS_OFF)	Do not use database field widths
Col	0	Assign the name ON to the first column and specify the header text
ColName	"ON"	
ColHeaderText	"Owner Name"	

Col	1	Assign the name AD to the second column and specify the header text
ColName	"AD"	
ColHeaderText	"Address"	
Col	2	Assign the name PH to the third column and specify the header text
ColName	"PH"	
ColHeaderText	"Phone"	
Col	3	Assign the name BR to the fourth column and specify the header text
ColName	"BR"	
ColHeaderText	"Breed"	
Col	4	Assign the name DN to the fifth column and specify the header text
ColName	"DN"	
ColHeaderText	"Name"	
Col	5	Assign the name CL to the sixth column and specify the header text
ColName	"CL"	
ColHeaderText	"Color"	
Col	6	Assign the name SI to the seventh column and specify the header text
ColName	"SI"	
ColHeaderText	"Sire"	
Col	7	Assign the name DA to the eighth column and specify the header text
ColName	"DA"	
ColHeaderText	"Dam"	



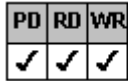
[Referencing a Column](#)
[Creating Column Headers](#)
[Binding Columns to Fields in a Database](#)

9. Make the columns children of groups and position the columns within the groups by setting the [ColParentGroup](#), [ColPosInParent](#), [ColLevel](#), and [ColLevelHeight](#) properties as follows. Use the [ColFromName](#) property to reference the columns.

Property	Setting	What this means
ColFromName	"ON"	Make the Owner column a child of the first group (Owner Info) and position the column in the group.
ColParentGroup	0	
ColPosInParent	0	
ColFromName	"AD"	Make the Address column a child of the first group (Owner Info) and position the column in the group. Start the column on level 1 and make it two levels high.
ColParentGroup	0	
ColPosInParent	1	
ColLevel	1	
ColLevelHeight	2	
ColFromName	"PH"	Make the Phone column a child of the first group (Owner Info) and position the column in the group. Display the column on level 0.
ColParentGroup	0	
ColPosInParent	2	
ColLevel	0	
ColFromName	"BR"	Make the Breed column a child of the second group (Registrant) and position the column in the group. Make the column two levels high.
ColParentGroup	1	
ColPosInParent	0	
ColLevelHeight	2	
ColFromName	"DN"	Make the Name column a child of the second group (Registrant) and position the column in the group. Display the column on level 0.
ColParentGroup	1	
ColPosInParent	1	
ColLevel	0	
ColFromName	"CL"	Make the Color column a child of the second group (Registrant) and position the column in the group. Display the column on level 1.
ColParentGroup	1	
ColPosInParent	2	
ColLevel	1	

ColFromName	"SI"	Make the Sire column a child of the third group
ColParentGroup	2	(Lineage) and position the column in the group.
ColPosInParent	0	

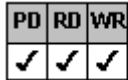
ColFromName	"DA"	Make the Dam column a child of the third group
ColParentGroup	2	(Lineage) and position the column in the group.
ColPosInParent	1	Display the column on level 1.
ColLevel	1	



[Referencing a Column](#)
[Making a Column a Child of a Group](#)
[Defining the Position of a Column Within the Control](#)
[Creating Levels of Columns Within a Row](#)

10. Display text in multiple lines in the Address and Breed columns by setting the [ListApplyTo](#) and [MultiLine](#) properties as follows. Use the [ColFromName](#) property to reference the columns.

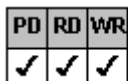
Property	Setting (Constant)	What this means
ListApplyTo	12 (LC_LISTAPPLYTO_SINGLE_ITEM)	Apply the multiple line property setting to the Address column
ColFromName	"AD"	
MultiLine	2 (LC_MULTILINE_MULTI_LINE)	
ListApplyTo	12 (LC_LISTAPPLYTO_SINGLE_ITEM)	Apply the multiple line property setting to the Breed column
ColFromName	"BR"	
MultiLine	2 (LC_MULTILINE_MULTI_LINE)	



[Applying Properties to Specific Parts of the Control](#)
[Referencing a Column](#)
[Wrapping Text in a List Pro Control](#)

11. Define the background and text colors for the column headers, group headers, and selected rows by setting the [ListApplyTo](#), [BackColor](#), and [ForeColor](#) properties as follows.

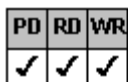
Property	Setting (Constant)	What this means
ListApplyTo	7 (LC_LISTAPPLYTO_COL_HEADERS)	Display black background and white text colors in the column headers
BackColor	RGB (0, 0, 0) 00000000	
ForeColor	RGB (255, 255, 255) 00FFFFFF	
ListApplyTo	8 (LC_LISTAPPLYTO_GROUP_HEADERS)	Display green background color in the group headers
BackColor	RGB (0, 255, 0) 0000FF00	
ListApplyTo	4 (LC_LISTAPPLYTO_SEL_ROWS)	Display red background and white text color to selected rows
BackColor	RGB (255, 0, 0) 000000FF	
ForeColor	RGB(255, 255, 255) 00FFFFFF	



[Applying Properties to Specific Parts of the Control](#)
[Changing Text Color and Fonts](#)
[Changing the Background Color](#)

12. Display a flat, green line between the rows by setting the [ListApplyTo](#), [LineApplyTo](#), [LineStyle](#), and [LineColor](#) properties as follows.

Property	Setting (Constant)	What this means
ListApplyTo	1 (LC_LISTAPPLYTO_ALL_ROWS)	Display green, flat lines between all rows
LineApplyTo	1 (LC_LINEAPPLYTO_ROWS)	
LineStyle	2 (LC_LINESTYLE_FLAT)	
LineColor	RGB (0, 255, 0) 0000FF00	
ListApplyTo	0 (LC_LISTAPPLYTO_DEFAULT)	Reset ListApplyTo property



[Applying Properties to Specific Parts of the Control](#)
[Adding Lines](#)
[Customizing Lines](#)

13. Run the project.

The list box should look similar to the following figure, depending on the size of your list box. Notice that the Phone, Name (Registrant), and Color columns are too small.

Owner Info		Registrant		Lineage
Owner Name	Phone	Breed	Name	Sire
Address			Color	Dam
Julia James	(919) 5	Airedale Terrier	BJ's Amelia	Bark Vader
5008 Monitor Dr., Peak NC, 27052			Black/Tan	Moorman's Candy Queen
Tim Wrenn	(201) 5	Grand Griffon Vendéen	Roy Du Roi	Hunter's Best
321 Webster Street, Queens NY, 10163			White/Orange	Petite Pauline

14. If necessary, resize your control to display all groups and columns as shown in step 14 and rerun the project.
15. Resize the Phone, Name (Registrant), and Color columns by completing the following steps:
 - a. Move the mouse pointer over the column header border between the Name (Owner Info) and Phone columns until the pointer changes to a resize icon. Drag this border to the left until the entire phone number is displayed in the Phone column.
 - b. Move the mouse pointer over the right column header border of the Breed column until the pointer changes to a resize icon. Drag this border to the left until the entire Name is displayed in the Name column.

Because this is a common border with the Color column header, the Color column also resizes.

The list box should appear similar to the following figure:

Owner Info		Registrant		Lineage
Owner Name	Phone	Breed	Name	Sire
Address			Color	Dam
Julia James	(919) 555-0079	Airedale Terrier	BJ's Amelia Airehart Airedale	Bark Vader
5008 Monitor Dr., Peak NC, 27052			Black/Tan	Moorman's Candy Queen
Tim Wrenn	(201) 555-7345	Grand Griffon Vendéen	Roy Du Roi	Hunter's Best
321 Webster Street, Queens NY, 10163			White/Orange	Petite Pauline

Step 15 illustrates how to change column widths using the mouse. To permanently change the column widths, you would have to set the column widths in code at run time.

[Print](#)

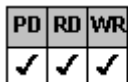
[Copy](#)

[Close](#)

To customize the list box

Property Designer

1. Allow resizing of the columns and groups by setting the [AllowColResize](#) and [AllowGrpResize](#) properties.
 - a. On the [General subtab of the Columns designer page](#),
 - i. Select the AllowColResize property from the properties list box.
 - ii. Select the 2 - Resize Col or Header option button under AllowColResize in the property value area.
 - b. On the [General subtab of the Groups designer page](#),
 - i. Select the AllowGrpResize property from the properties list box.
 - ii. Select the 2 - Resize Grp or Header option button under AllowGrpResize in the property value area.

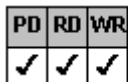


[Using the Mouse to Resize Columns](#)
[Using the Mouse to Resize Groups](#)

2. Change the background color of the list box by setting the [BackColor](#) property.

On the [List subtab of the ApplyTo designer page](#),

 - a. Select 0 - Default (All) from the List Apply To drop-down list box.
 - b. Select the BackColor property from the properties list box.
 - c. Choose the Color button under BackColor in the property value area to display the BackColor dialog box.
 - d. Select the light gray color on the color chart.
 - e. Choose the OK button.

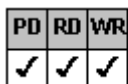


[Changing the Background Color](#)

3. Create a lowered, three-dimensional appearance for lines by setting the [LineStyle](#) property.

On the [Line subtab of the ApplyTo designer page](#),

 - a. Select 0 - Default (All) from the List Apply To drop-down list box.
 - b. Select 0 - Default from the Line Apply To drop-down list box.
 - c. Select the LineStyle property from the properties list box.
 - d. Select the 3 - Lowered option button under LineStyle in the property value area.

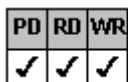


[Customizing Lines](#)

4. Display the column headers and create three column levels by setting the [ColumnHeaderShow](#) and [ColumnLevels](#) properties.

On the [General subtab of the Columns designer page](#),

 - a. Select the ColumnHeaderShow property from the properties list box.
 - b. Select the True option button under ColumnHeaderShow in the property value area.
 - c. Select the ColumnLevels property from the properties list box.
 - d. Type 3 in the box under ColumnLevels in the property value area.



[Creating Multiple Columns](#)
[Creating Column Headers](#)
[Creating Levels of Columns Within a Row](#)

5. Create the three groups and display the group headers by setting the [Groups](#) and [GroupHeaderShow](#) properties.
 - a. On the [Specific subtab of the Groups designer page](#),

- i. Select the Groups property from the properties list box.
- ii. Type 3 in the box under Groups in the property value area.
- b. On the [General subtab of the Groups designer page](#),
 - i. Select the GroupHeaderShow property from the properties list box.
 - ii. Select the True option button under GroupHeaderShow in the property value area.

PD	RD	WR
✓	✓	✓

[Creating Groups](#)
[Creating Group Headers](#)

- 6. Specify pixels as the unit of measurement for the group widths by setting the [ColumnWidthScale](#) property. On the [General subtab of the Columns designer page](#),
 - a. Select the ColumnWidthScale property from the properties list box.
 - b. Select the 1 - Pixels option button under ColumnWidthScale in the property value area.

PD	RD	WR
✓	✓	✓

[Specifying the Group Width](#)

- 7. Define the name, width, and header text for each group by setting the [Grp](#), [GrpName](#), [GrpHeaderText](#), and [GrpWidth](#) properties. On the [Specific subtab of the Groups designer page](#),
 - a. Either select 0 from the Grp drop-down list box under Individual or click the first group in the preview area.
 - b. Select the GrpName property from the properties list box.
 - c. Type OI in the box under GrpName in the property value area.
 - d. Select the GrpHeaderText property from the properties list box.
 - e. Type Owner Info in the box under GrpHeaderText in the property value area.
 - f. Select the GrpWidth property from the properties list box.
 - g. Type 170 in the box under GrpWidth in the property value area.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

- h. Repeat steps 7.a-7.g for the remaining two groups using the following data.

Group number	Group name	Header text	Group Width
1	RG	Registrant	225
2	LN	Lineage	125

PD	RD	WR
✓	✓	✓

[Referencing a Group](#)
[Creating Group Headers](#)
[Specifying the Group Width](#)

- 8. Define the column names and custom header text by setting the [DataAutoHeadings](#), [DataAutoSizeCols](#), [Col](#), [ColName](#), and [ColHeaderText](#) properties.
 - a. On the [General subtab of the Data Binding designer page](#),
 - i. Select the DataAutoHeadings property from the properties list box.
 - ii. Select the False option button under DataAutoHeadings in the property value area.
 - iii. Select the DataAutoSizeCols property from the properties list box.
 - iv. Select the 0 - Off option button under DataAutoSizeCols in the property value area.
 - b. On the [Specific subtab of the Columns designer page](#),
 - i. Either select 0 from the Col drop-down list box under Individual or click the first column in the preview area.
 - ii. Select the ColName property from the properties list box.
 - iii. Type ON in the box under ColName in the property value area.
 - iv. Select the ColHeaderText property in the properties list box.

- v. Type Owner Name in the box under ColHeaderText in the property value area.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

- vi. Repeat step 8.b.i 8.b.v for the remaining seven columns using the following data.

Column number	Column name	Header text
1	AD	Address
2	PH	Phone
3	BR	Breed
4	DN	Name
5	CL	Color
6	SI	Sire
7	DA	Dam

PD	RD	WR
✓	✓	✓

[Referencing a Column](#)
[Creating Column Headers](#)
[Binding Columns to Fields in a Database](#)

9. Make the columns children of groups and position the columns within the groups by setting the [ColParentGroup](#), [ColPosInParent](#), [ColLevel](#), and [ColLevelHeight](#) properties. Use the column name to reference the columns.

On the [Specific subtab of the Columns designer page](#),

- a. Make the Owner Name column a child of the first group (Owner Info) and position the column in the group.
 - i. Either select ON from the Col Name drop-down list box or click the Owner Name column in the preview area.
 - ii. Select the ColParentGroup property from the properties list box.
 - iii. Type 0 in the box under ColParentGroup in the property value area.
 - iv. Select the ColPosInParent property from the properties list box.
 - v. Type 0 in the box under ColPosInParent in the properties value area.
- b. Make the Address column a child of the first group (Owner Info) and position the column in the group. Start the column on level 1 and make it two levels high.
 - i. Either select AD from the Col Name drop-down list box or click the Address column in the preview area.
 - ii. Select the ColParentGroup property from the properties list box.
 - iii. Type 0 in the box under ColParentGroup in the property value area.
 - iv. Select the ColPosInParent property from the properties list box.
 - v. Type 1 in the box under ColPosInParent in the properties value area.
 - vi. Select the ColLevel property from the properties list box.
 - vii. Type 1 in the box under ColLevel in the property value area.
 - viii. Select the ColLevelHeight property from the properties list box.
 - ix. Type 2 in the box under ColLevelHeight in the property value area.
- c. Make the Phone column a child of the first group (Owner Info) and position the column in the group. Display the column on level 0.
 - i. Either select PH from the Col Name drop-down list box or click the Phone column in the preview area.
 - ii. Select the ColParentGroup property from the properties list box.
 - iii. Type 0 in the box under ColParentGroup in the property value area.
 - iv. Select the ColPosInParent property from the properties list box.
 - v. Type 2 in the box under ColPosInParent in the properties value area.
 - vi. Select the ColLevel property from the properties list box.
 - vii. Type 0 in the box under ColLevel in the property value area.
- d. Make the Breed column a child of the second group (Registrant) and position the column in the group. Make the column two levels high.
 - i. Either select BR from the Col Name drop-down list box or click the Breed column in the preview area.
 - ii. Select the ColParentGroup property from the properties list box.

- iii. Type 1 in the box under ColParentGroup in the property value area.
 - iv. Select the ColPosInParent property from the properties list box.
 - v. Type 0 in the box under ColPosInParent in the properties value area.
 - vi. Select the ColLevelHeight property from the properties list box.
 - vii. Type 2 in the box under ColLevelHeight in the property value area.
- e. Make the Name column a child of the second group (Registrant) and position the column in the group. Display the column on level 0.
- i. Either select DN from the Col Name drop-down list box or click the Name column in the preview area.
 - ii. Select the ColParentGroup property from the properties list box.
 - iii. Type 1 in the box under ColParentGroup in the property value area.
 - iv. Select the ColPosInParent property from the properties list box.
 - v. Type 1 in the box under ColPosInParent in the properties value area.
 - vi. Select the ColLevel property from the properties list box.
 - vii. Type 0 in the box under ColLevel in the property value area.
- f. Make the Color column a child of the second group (Registrant) and position the column in the group. Display the column on level 1.
- i. Either select CL from the Col Name drop-down list box or click the Color column in the preview area.
 - ii. Select the ColParentGroup property from the properties list box.
 - iii. Type 1 in the box under ColParentGroup in the property value area.
 - iv. Select the ColPosInParent property from the properties list box.
 - v. Type 2 in the box under ColPosInParent in the properties value area.
 - vi. Select the ColLevel property from the properties list box.
 - vii. Type 1 in the box under ColLevel in the property value area.
- g. Make the Sire column a child of the third group (Lineage) and position the column in the group.
- i. Either select SI from the Col Name drop-down list box or click the Sire column in the preview area.
 - ii. Select the ColParentGroup property from the properties list box.
 - iii. Type 2 in the box under ColParentGroup in the property value area.
 - iv. Select the ColPosInParent property from the properties list box.
 - v. Type 0 in the box under ColPosInParent in the properties value area.
- h. Make the Dam column a child of the third group (Lineage) and position the column in the group. Display the column on level 1.
- i. Either select DA from the Col Name drop-down list box or click the Dam column in the preview area.
 - ii. Select the ColParentGroup property from the properties list box.
 - iii. Type 2 in the box under ColParentGroup in the property value area.
 - iv. Select the ColPosInParent property from the properties list box.
 - v. Type 1 in the box under ColPosInParent in the properties value area.
 - vi. Select the ColLevel property from the properties list box.
 - vii. Type 1 in the box under ColLevel in the property value area.

PD	RD	WR
✓	✓	✓

[Referencing a Column](#)
[Making a Column a Child of a Group](#)
[Defining the Position of a Column Within the Control](#)
[Creating Levels of Columns Within a Row](#)

10. Display text in multiple lines in the Address and Breed columns by setting the [ListApplyTo](#) and [MultiLine](#) properties. Use the column name to reference the columns.
- On the [List subtab of the ApplyTo designer page](#),
- a. Select 12 - Single Item from the List Apply To drop-down list box.

- b. Select AD from the Col Name drop-down list box.
- c. Select the MultiLine property from the properties list box.
- d. Select the 2 - Multiple Line option button under MultiLine in the property value area.
- e. Select 12 - Single Item from the List Apply To drop-down list box.
- f. Select BR from the Col Name drop-down list box.
- g. Select the MultiLine property from the properties list box.
- h. Select the 2 - Multiple Line option button under MultiLine in the property value area.

PD	RD	WR
✓	✓	✓

[Applying Properties to Specific Parts of the Control](#)
[Referencing a Column](#)
[Wrapping Text in a List Pro Control](#)

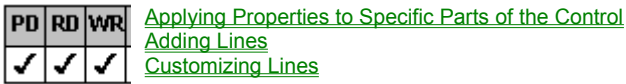
11. Define the background and text colors for the column headers, group headers, and selected rows by setting the [ListApplyTo](#), [BackColor](#), and [ForeColor](#) properties.
 - a. Display black background and white text colors in the column headers.
 On the [List subtab of the ApplyTo designer page](#).
 - i. Select 7 - Col Headers from the List Apply To drop-down list box.
 - ii. Select the BackColor property from the properties list box.
 - iii. Choose the Color button under BackColor in the property value area to display the BackColor dialog box.
 - iv. Select the black color on the color chart.
 - v. Choose the OK button.
 - vi. Select the ForeColor property from the properties list box.
 - vii. Choose the Color button under ForeColor in the property value area to display the ForeColor dialog box.
 - viii. Select the white color on the color chart.
 - ix. Choose the OK button.
 - b. Display green background color in the group headers.
 - i. Select 8 - Group Headers from the List Apply To drop-down list box.
 - ii. Select the BackColor property.
 - iii. Choose the Color button under BackColor in the property value area to display the BackColor dialog box.
 - iv. Select the green color on the color chart.
 - v. Choose the OK button.
 - c. Display red background and white text color to selected rows.
 - i. Select 4 - Sel Rows from the List Apply To drop-down list box.
 - ii. Select the BackColor property from the properties list box.
 - iii. Choose the Color button under BackColor in the property value area to display the BackColor dialog box.
 - iv. Select the red color on the color chart.
 - v. Choose the OK button.
 - vi. Select the ForeColor property from the properties list box.
 - vii. Choose the Color button under ForeColor in the property value area to display the ForeColor dialog box.
 - viii. Select the white color on the color chart.
 - ix. Choose the OK button.

PD	RD	WR
✓	✓	✓

[Applying Properties to Specific Parts of the Control](#)
[Changing Text Color and Fonts](#)
[Changing the Background Color](#)

12. Display a flat, green line between the rows by setting the [ListApplyTo](#), [LineApplyTo](#), [LineStyle](#), and [LineColor](#) properties.
 On the [Line subtab of the ApplyTo designer page](#).
 - a. Select 1 - All Rows from the List Apply To drop-down list box.

- b. Select 1 - Rows from the Line Apply To drop-down list box.
- c. Select the LineStyle property from the properties list box.
- d. Select the 2 - Flat option button under LineStyle in the property value area.
- e. Select the LineColor property from the properties list box.
- f. Choose the Color button under LineColor in the property value area to display the LineColor dialog box.
- g. Select the green color on the color chart.
- h. Choose the OK button.



13. Choose the OK button to close the FarPoint Property Designer and apply the changes to the control on your form.
14. Run the project.

The list box should look similar to the following figure, depending on the size of your list box. Notice that the Phone, Name (Registrant), and Color columns are too small.

Owner Info		Registrant		Lineage
Owner Name	Phone	Breed	Name	Sire
Address			Color	Dam
Julia James	(919) 5	Airedale Terrier	BJ's Amelia	Bark Vader
5008 Monitor Dr., Peak NC, 27052			Black/Tan	Moorman's Candy Queen
Tim Wrenn	(201) 5	Grand Griffon Vendéen	Roy Du Roi	Hunter's Best
321 Webster Street, Queens NY, 10163			White/Orange	Petite Pauline

15. If necessary, resize your control to display all groups and columns as shown in step 14 and rerun the project.
16. Resize the Phone, Name (Registrant), and Color columns by completing the following steps:
 - a. Move the mouse pointer over the column header border between the Name (Owner Info) and Phone columns until the pointer changes to a resize icon. Drag this border to the left until the entire phone number is displayed in the Phone column.
 - b. Move the mouse pointer over the right column header border of the Breed column until the pointer changes to a resize icon. Drag this border to the left until the entire Name is displayed in the Name column.

Because this is a common border with the Color column header, the Color column also resizes.

The list box should appear similar to the following figure:

Owner Info		Registrant		Lineage
Owner Name	Phone	Breed	Name	Sire
Address			Color	Dam
Julia James	(919) 555-0079	Airedale Terrier	BJ's Amelia Airehart Airedale	Bark Vader
5008 Monitor Dr., Peak NC, 27052			Black/Tan	Moorman's Candy Queen
Tim Wrenn	(201) 555-7345	Grand Griffon Vendéen	Roy Du Roi	Hunter's Best
321 Webster Street, Queens NY, 10163			White/Orange	Petite Pauline

Step 16 illustrates how to change column widths using the mouse. To permanently change the column widths, you would have to set the column widths in code at run time.

Using ActiveX Controls

ActiveX controls are OLE 2 custom controls. You can use the fpCombo and fpList ActiveX controls in most development environments that support ActiveX controls.

ActiveX controls are designed to function more independently than VBX and DLL controls. That is, they are designed to be independent objects that you can easily use in many different types of containers and environments. Designed to present a consistent interface, ActiveX controls and their containers provide a unique means for interaction.

The following topics provide an overview of some ActiveX-specific concepts and features. Throughout the help, ActiveX-specific information has been included to further acquaint you with the fpCombo and fpList ActiveX controls and to provide instructions for their use.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Types of ActiveX Properties](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Setting Properties](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Using Events](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Summary of ActiveX Control Features](#)

Types of ActiveX Properties

ActiveX controls can support four types of properties: stock, standard extender, custom, and ambient.

Stock properties are always provided by containers that support ActiveX controls. Examples of stock properties are the [BackColor](#) and Enabled properties.

Standard extender properties can be, but are not necessarily, provided by containers that support ActiveX controls. Examples of standard extender properties include the Height and DataChanged properties.

Custom properties are properties added by the controls designer to enhance certain characteristics of the ActiveX control. For example, the [Appearance](#) and [ThreeDFrameColor](#) properties are custom properties.

Ambient properties are properties that let the ActiveX control respond to characteristics of its container. For example, the AmbientScaleUnit property lets the ActiveX control use the same scale unit as its container. Note that containers need not support ambient properties.

The property topics in the property reference provide information about the ActiveX type for the fpCombo and fpList controls properties. Each property that is an ActiveX stock or standard extender property is noted as such. If no information is provided for a given property, that property is a custom property.

Setting Properties

ActiveX controls provide multiple avenues for viewing and setting control properties. You can view and set properties in the FarPoint Property Designer pages, which are organized by control characteristics. For more information on the designer pages, see [Using the FarPoint Property Designer](#).

You can also view all the design-time properties available for the control in an alphabetical list called a property browser. For example, in Visual Basic the Properties window provides a list that lets you view and set properties. In addition, you can view the available settings for enumerated properties. However, different containers provide different browser interfaces.

Finally, you can set properties in code at run time. The type of code you use and the syntax depends on your container.

Using Events

ActiveX controls can support three types of events: stock, standard extender, and custom.

Stock events are provided by all ActiveX containers. Examples of stock events include the Click event, the DbClick_event, and the KeyPress event.

Standard extender events are events that ActiveX containers can, but do not necessarily, provide. Examples of standard extender events include the DragDrop event and the GotFocus event.

Custom events are created by the controls designer to support certain actions users perform on that control. For example, the fpCombo control includes the [ColWidthChange](#) event, which occurs when the user resizes the column width with the mouse.

Summary of ActiveX Control Features

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

List Pro ActiveX controls are supported by most OLE-compliant containers that support ActiveX controls.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ActiveX controls can have four types of properties: stock, standard extender, ambient, and custom.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ActiveX controls provide several methods of interaction: the FarPoint Property Designer, the standard property browser, and run-time settings.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ActiveX controls can have three types of events: stock, standard extender, and custom.

Using DLL Controls

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Working with Libraries](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating the fpCombo and fpList Controls](#)

■ [Getting and Setting Property Values](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Working with String Data](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Receiving Notifications](#)

Working with Libraries

You can use the List Pro DLL controls in the C programming language using the provided dynamic-link library (.DLL) files. You can also use the .DLL files in the C++ programming language.

The following topics describe how to load and unload DLL libraries.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Loading the Library](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Unloading the Library](#)

Loading the Library

To use the List Pro controls with C, you must first load the .DLL files for these controls. To load a library, you must declare a variable of type HINSTANCE and then use a Windows API call. The Windows API call loads the controls .DLL files into memory and returns the handle.

The following code loads the List Pro DLL:

```
HINSTANCE hInstLPro;  
hInstLPro = LoadLibrary("listpro_filename.DLL");
```

You must also link the static link library (.LIB) with your application.

Unloading the Library

Before you exit your application, you should unload the DLL library. Unload .DLL files using the `FreeLibrary` command, which is a Windows API call. Specify the .DLL file by using the assigned identifier from the `LoadLibrary` function. In the previous code example, the variable used for the .DLL file is `hInstLPro`.

The following code frees the DLL from memory:

```
FreeLibrary(hInstLPro);
```

Creating the fpCombo and fpList Controls

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating Controls With a CreateWindow Function](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating Controls With a CONTROL Statement](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating Controls in MFC Using the Create Member Function](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating Controls in OWL Using the New Operator](#)

Creating Controls with the CreateWindow Function

In C or C++, you can use the Windows CreateWindow function to create the control. The CreateWindow function specifies the window class, the window title, the window style, the initial position and size of the window, the windows parent, and the menu.

The following code creates an fpCombo control:

```
HWND hWndCombo;  
hWndCombo = CreateWindow(  
    FPCLASS_COMBO,           // window class name  
    "",                      // window caption  
    WS_CHILD | WS_VISIBLE   // window style  
    Rect.left,              // initial x position  
    Rect.top,               // initial y position  
    Rect.right - Rect.left, // initial x size  
    Rect.bottom - Rect.top, // initial y size  
    hWnd,                  // parent window handle  
    IDC_COMBO,             // window ID  
    hInstance,             // program instance handle  
    NULL,                  // creation parameter  
);
```

The CreateWindow function creates controls one at a time. Use the AppStudio™ in Visual C++™ or the Resource Workshop in Borland® C++ to create a control that you can bring up on a dialog.

Creating Controls with a CONTROL Statement

In C or C++, you can create List Pro controls using a CONTROL statement in a resource file in a dialog box. The CONTROL statement defines a control by using arguments separated by commas. These arguments specify the class of the control, give the control an identifier, define the control style, and define the position and dimensions of the control within the parent dialog.

The following statement creates an fpList control:

```
CONTROL "", IDC_LIST, FPCLASS_LIST, WS_CHILD | WS_VISIBLE |
```

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

```
WS_TABSTOP, 5, 35, 165, 60
```

To create an fpCombo control, you would specify the window class name for the fpCombo control, FPCLASS_COMBO, and specify a window identifier, IDC_COMBO, for example.

Creating Controls in MFC Using the Create Member Function

If you are using MFC, you can create controls in code using the Create member function of the class for the control you are creating. To do so, declare an object or pointer variable, then call the Create member function. The following code illustrates how you might create an fpList control. For this example, it is assumed that the control is created in a parent window class.

```
#define ID_LIST 1000
```

```
RECT rect = {0, 0, 100, 100};
```

```
CfpList Lst;
```

```
Lst.Create(WS_VISIBLE|WS_CHILD, &rect, this, ID_LIST);
```

The parameters of the Create member function let you specify the styles for the control, the coordinates for the control, the parent of the control, and a unique identifier.

Creating Controls in OWL Using the New Operator

If you are using OWL, you can create controls in code using a constructor. To do so, declare a pointer variable, then use the New operator to call the constructor. The following code illustrates how you might create an fpList control. For this example, it is assumed that the control is created in a parent window class.

```
#define ID_LIST 1000
```

```
short left = 0;  
short top = 0;  
short width = 100;  
short height = 100;
```

```
CfpList* Lst;
```

```
Lst = new CfpList(this, ID_LIST, left, top, width, height);
```

The parameters of the constructor let you specify the parent of the control, a unique identifier, the left coordinate, the top coordinate, and the width and height of the control.

Getting and Setting Property Values

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Overview](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Get Function](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Set Function](#)

Overview

Once you have loaded the .DLL file and created a control, you can use the specific Get and Set functions for each property to retrieve and set property values in your application.

Use the Get function to retrieve property values in your application. For example, to set the value of a variable equal to the value of the [BorderStyle](#) property for an fpCombo control, you must return the value of the BorderStyle property. The following C code assigns the value of the parameter lpvalue to the variable Bstyle.

```
Bstyle = LC_GetBorderStyle(hWnd, lpvalue);
```

Use the Set function to set property values in your application. For example, to set the value of the BorderStyle property for an fpCombo control to rounded, use the following code:

```
LC_SetBorderStyle(hWnd, LC_BORDERSTYLE_ROUNDED);
```

Note Write-only properties do not have corresponding Get functions, and read-only properties do not have corresponding Set functions. Data-aware properties, such as the DataField and DataSync properties, are not available for the DLL controls.

The DLL Get and Set functions are declared in the header files that accompany this product. The header files are installed when you install List Pro using the installation program. The following topics describe the DLL functions, including the function declarations and function calls. The declarations are included for descriptive purposes; you do not need to declare the functions in your application if you have included the correct header files.

The following topics explain the components of the Get and Set functions in detail.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Get Function](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Set Function](#)

Get Function

The Get function retrieves the current value of a specified property. Each Get function uses the same pattern in code. The parameters are the controls window handle and a pointer to the variable that contains the resulting value.

The following example is for an fpCombo Get function declaration; the fpList control uses the same pattern.

```
int LC_GetListIndex (HWND hWnd, long FAR *lpvalue);
```

Return Value

The Get functions return a value of 0 if the function is successful or nonzero if an error has occurred.

Function Name

Most properties that can be read in the List Pro controls have their own Get functions in C and C++. Function names that are unique for each control begin with "Cbx" or "Lbx", followed by the word "Get" and the property name. Therefore, the Get function for the MaxDrop property is named the "CbxGetMaxDrop" function, while the Get function for the [MultiSelect](#) property is named the "LbxGetMultiSelect" function. Functions that are common to both controls begin with "LC_", followed by the word "Get" and the property name. Therefore the Get function for the NoIntegralHeight property is named the "LC_GetNoIntegralHeight" function.

The following code declares the Get function for the BorderWidth property:

```
int LC_GetBorderWidth(HWND hWnd, int FAR *lpvalue);
```

In the example, "LC_GetBorderWidth" is the name of the function. The [BorderWidth](#) property is an Integer data type, which is indicated by "int" before the *lpValue parameter.

Data Type

The Visual Basic data type for each property is listed in the online *Reference Guide*. The following table lists the Visual Basic data types and the corresponding DLL data types.

Visual Basic data type	Corresponding DLL data type
------------------------	-----------------------------

Color	COLORREF
-------	----------

Integer	int
---------	-----

Integer (Boolean)	BOOL
-------------------	------

Integer (Enumerated)	int
----------------------	-----

Long Integer	long
--------------	------

Picture	FP_PICTITEM
---------	-------------

Single	int, float
--------	------------

Note Single data types that are used as coordinates are computed in pixels in the DLL version, so the data type is an integer value.

String	LPSTR
--------	-------

Note Parameters of this data type are actually pointers to buffers. See [Working with String Data](#) for more information.

The following Get function declaration describes the data type of the return value and the parameters required by the fpCombo GetListDown function:

```
int CbxGetListDown(HWND hWnd, BOOL FAR *lpvalue);
```

The following function call illustrates how you might use the fpCombo GetListDown function in code, assigning the return value of the function to the variable isldown:

```
int iRet;  
BOOL isldown;  
iRet = CbxGetListDown(hWnd, &isldown);
```

The function declaration lets you know to define the isldown variable as a boolean data type and to pass the fpCombo controls window handle to the function.

Set Function

The Set function call sets a new value for a specific property. Each Set function uses the same pattern in code. This code pattern specifies the data type of the return value, the case-sensitive name of the function, and two function parameters: the window handle of the control and the new value for the property.

The following example is for an fpCombo Set function call; the fpList control uses the same pattern.

```
int LC_SetListIndex (HWND hWnd, long value);
```

Return Value

The Set functions return a value of 0 if the function is successful or nonzero if an invalid parameter is passed to the Set function.

Function Name

Most List Pro properties that can be written have their own Set function in C or C++. Function names that are unique for each control begin with "Cbx" or "Lbx", followed by the word "Set" and the property name. Therefore, the Set function for the Style property is named the "CbxSetStyle" function, while the Set function for the [SelMax](#) property is named the "LbxSetSelMax" function. Functions that are common to both controls begin with "LC_", followed by the word "Set" and the property name. Therefore, the Set function for the [NoIntegralHeight](#) property is named the "LC_SetNoIntegralHeight" function.

The following code declares the SetLineWidth function:

```
int LC_SetLineWidth(HWND hWnd, int value);
```

In the example, "LC_SetLineWidth" is the name of the function. The SetLineWidth function requires an Integer data type, which is indicated by "int" before the value parameter.

Data Value

You must indicate the new value for each Set function as a parameter of the function. The new value must correspond to the appropriate data type for the function.

The following code declares the SetSearchText function. The declaration indicates that the new value must be a string.

```
int LC_SetSearchText(HWND hWnd, LPCSTR value);
```

When you are setting the value for the SetSearchText function, you must pass the appropriate string value. The following function call sets the value of the SetSearchText function to "Cancel":

```
LC_SetSearchText(hWnd, "Cancel");
```

Working with String Data

When you use the Get or Set function for a string value, you must supply a buffer to indicate the array of available characters for the string, as in the following code:

```
char szbuffer[#]
```

Replace the # character with the number of characters available in the array. For example, the SearchText function for the fpCombo control requires a string value. The first statement in the following example indicates the buffer can contain up to 39 characters (with an additional character for the Null terminator).

```
char szBuffer[40];
LC_GetSearchText(hWnd, szBuffer, 39);
LC_SetSearchText(hWnd, "I am the caption");
```

If you do not know the exact length of the string, use code similar to the following example.

```
HANDLE hBuffer;
int nLen;
int rCode;
rCode = LC_GetTextLength(hWnd, &nLen);
If (hBuffer = GlobalAlloc(GHND, nLen + 1))
{
    LPSTR lpszBuffer = GlobalLock (hBuffer);
    LC_GetSearchText(hWnd, lpszBuffer, nLen);
    .
    .
    (your code)
    .
    GlobalUnlock(hBuffer);
    GlobalFree(hBuffer);
}
```

Receiving Notifications

For the fpCombo or fpList control, you must include the LBS_NOTIFY style in the RC file to receive all notifications.

Getting Started

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Parts of a List Pro Control](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Applying Properties to Specific Parts of the Control](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

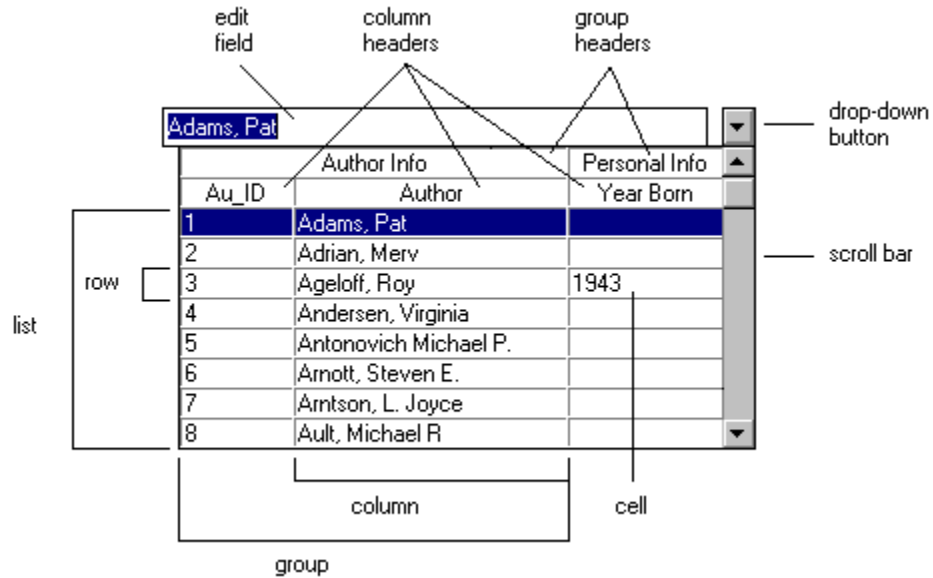
[Setting Up the Control](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Choosing the fpCombo Control Style](#)

Parts of a List Pro Control

The following figure shows the basic parts of a List Pro control, illustrated using a drop-down combo box.



Different control types might not have some of these features. For example, a simple combo style combo box does not have a drop-down button.

Applying Properties to Specific Parts of the Control

You can customize the appearance of any of the following parts of a List Pro control:

All rows	All columns	All groups
Single row	Single column	Single group
Odd-numbered rows	All column headers	All group headers
Even-numbered rows	Single column header	Single group header
Selected rows	Single cell	

The properties you can apply to these different parts of the control are called designated-list properties. Use the [ListApplyTo](#) property to specify where the following designated-list properties apply:

AlignH	ForeColor	List3DTextHighlightColor
AlignV	Line3DDark	List3DTextOffset
BackColor	Line3DLight	List3DTextShadowColor
Font	Line3DWidth	MultiLine
FontBold	LineApplyTo	Picture
FontItalic	LineStyle	PictureAlignH
FontName	LineWidth	PictureAlignV
FontSize	List3DText	PictureSel
FontStrikethru	LineColor	TextOrientation
FontUnderline		

For many properties, you should specify to which part of the control you want the property applied. If you do not, the property is applied to the whole control.

Note that parts of the control by default inherit the property settings of other parts of the control. That is, if parts of the control do not have a specific setting, they look to the next highest level (hierarchical predecessor) to see what their settings should be. The hierarchy that determines this can be found in Appendix C, "Hierarchy of Property Settings" of the printed *List Pro User's Guide*.

For example, assume you have an fpList control with multiple columns and column headers and you have not set any properties for the column headers. If you set the MultiLine property for Column 1 to 2 (Multiple Line), all list items in Column 1 and the column header display text on multiple lines (individual column header is below an individual column in the hierarchical structure and therefore inherits the property setting for an individual column).

Setting Up the Control

A logical order for setting up a List Pro control is:

1. Choose the type of control (fpCombo or fpList).
2. If you want to link the control to a database, bind the control to the database.
For more information, see [Binding a Control to a Database](#)
3. If you chose an fpCombo control, choose the control style.
For more information, see [Choosing the fpCombo Control Style](#).
4. Set up your columns.
For more information, see [Working with Columns](#).
5. If you want groups, set up your groups.
For more information, see [Working with Groups](#).
6. Add list items and work with them.
For more information, see [Working with List Items](#).
7. Customize the appearance of the control or each individual part.
For more information, see [Customizing the Controls Appearance](#).

Choosing the fpCombo Control Style

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

An fpCombo control can have one of three styles: drop-down combo, simple combo, or drop-down list.

You can specify the offset of the list from the left side of the combo box. If the combo box has a drop-down combo or drop-down list style, you can specify the size of the gap between the edit field and the drop-down arrow. If the combo box has a drop-down combo or simple combo style, you can specify the maximum number of characters the user can enter in the edit field, and you can specify the color of the gray area surrounding the control.

For more information on each of the three combo box styles, see [Combo Box Styles](#).

[Print](#)

[Copy](#)

[Close](#)

To choose and customize an fpCombo control style

Designer Page

1. Specify the combo box style.
 - a. On the [Misc subtab of the Appearance designer page](#), select the [Style](#) property from the properties list box.
 - b. Select the appropriate option button under Style in the property value area.
2. Specify the offset of the list from the left side of the combo box.
 - a. Select the [ListLeftOffset](#) property from the properties list box.
 - b. Type the offset in the box under ListLeftOffset in the property value area.
3. If you specified 0 (Drop-Down Combo) or 2 (Drop-Down List) in step 1, define the gap size in pixels.
 - a. Select the [ComboGap](#) property from the properties list box.
 - b. Type the gap size in pixels in the box under ComboGap in the property value area.
4. If you specified 0 (Drop-Down Combo) or 1 (Simple Combo) in step 1, specify the maximum text length and the color of the gray area surrounding the control.
 - a. Select the [MaxEditLen](#) property from the properties list box.
 - b. Type the maximum text length in the box under MaxEditLen in the property value area.
 - c. On the [Color subtab of the Appearance designer page](#), select the [GrayAreaColor](#) property from the properties list box.
 - d. Choose the button under GrayAreaColor in the property value area to display the Gray Area Color dialog box.
 - e. Select a basic color or define your own custom color.
 - f. Choose the OK button.

[Print](#)

[Copy](#)

[Close](#)

To choose and customize an fpCombo control style

Browser/Code

1. Specify the combo box style with the [Style](#) property.
2. Specify the offset of the list from the left side of the combo box with the [ListLeftOffset](#) property.
3. If you specified 0 (Drop-Down Combo) or 2 (Drop-Down List) in step 1, define the gap size in pixels with the [ComboGap](#) property.
4. If you specified 0 (Drop-Down Combo) or 1 (Simple Combo) in step 1,
 - a. Specify the maximum text length with the [MaxEditLen](#) property.
 - b. Specify the color of the gray area surrounding the control with the [GrayAreaColor](#) property.

Working with List Items

For an overview on working with list items, see [List Items](#), [Data Binding](#), and [Virtual Mode](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Adding List Items](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Removing List Items](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Accessing List Items](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Sorting List Items](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Searching for List Items](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Customizing the Appearance of Selected Items](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Hiding the Focus Rectangle Around Selected Items](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Wrapping List Items in a Single-Column Control](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Working with Databases](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Using Virtual Mode](#)

Adding List Items

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To add items with the InsertRow property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To add a items to a single-column control with the List property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To add an item to a cell in a multiple-column control with the ColList property

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

You can add items (single-column) or rows of items (multiple-column) to the list in an fpCombo or fpList control in any of the following five ways:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

multiple-column control.

Use the [InsertRow](#) property to add an item to a single-column control or to add a row of items to a

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Use the [List](#) property to add data to a specific row in a single-column control.

You must specify the row for the list item.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Use the [ColList](#) property to add data to a specific cell in a multiple-column control.

You must specify the row and column for the list item.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Use the controls data binding capabilities to add items stored in a database table.

To bind the fpCombo or fpList control, see [Binding a Control to a Database](#) and [Binding Columns to Fields in a Database](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Use the AddItem method.

For information on the AddItem method, refer to the Visual Basic documentation.

[Print](#)

[Copy](#)

[Close](#)

To add items with the InsertRow property

Designer Page

1. Specify the row number at which the items or row of items should be inserted. On the [Add Data designer page](#), if you want

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

to insert rows at the end of the list, type 1 in the Row box; otherwise, either type the row number in the Row box or click the row in the preview area.

2. Add the items.
 - a. Select the [InsertRow](#) property in the properties list box.
 - b. Type the list item in the InsertRow box in the property value area.

If you are adding rows to a multiple-column fpCombo or fpList control, separate the column data with a vertical bar (|).

3. Repeat steps 1 and 2 until you have added all the items you want.

[Print](#)

[Copy](#)

[Close](#)

To add items with the InsertRow property

Code

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

1. Set the [Row](#) property to 1 if you want to insert rows at the end of the list; otherwise, designate where the new item or row of items is inserted with the Row property.

2. In an event procedure such as Form_Load, set the [InsertRow](#) property to the string you want to add.

If you are adding rows to a multiple-column fpCombo or fpList control, specify multiple items in the same row by joining them with a separator character. Use the [ColumnSeparatorChar](#) property to specify the separator character. The default separator character is a tab character (t or ASCII value 9).

3. Continue adding items by setting the InsertRow property.

[Print](#)

[Copy](#)

[Close](#)

To add items to a single-column control with the List property

Designer Page

1. Specify the row number at which the items or row of items should be inserted. On the [Add Data designer page](#), type

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

1 in the Row box if you want to insert rows at the end of the list; otherwise, either type the row number in the Row box or click the row in the preview area.

2. Add the item.
 - a. Select the [List](#) property in the properties list box.
 - b. Type the list item in the List box in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To add items to a single-column control with the List property

Code

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

1. Set the [Row](#) property to 1 if you want to insert rows at the end of the list; otherwise, designate where the new item or row of items is inserted with the Row property.
2. Specify the list item with the [List](#) property.

[Print](#)

[Copy](#)

[Close](#)

To add an item to a cell in a multiple-column control with the ColList property

Designer Page

1. Specify the column and row location. On the [Add Data designer page](#), either click the cell in the preview area or perform the following actions:
 - a. Select the column from either the Col, Col Name, or Col ID drop-down list box.
 - b. Type the row number in the Row box.
2. Add the list item.
 - a. Select the [ColList](#) property in the properties list box.
 - b. Type the list item in the ColList box in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To add an item to a cell in a multiple-column control with the ColList property

Code

1. Specify the column and row location with either the [Col](#), [ColFromID](#), or [ColFromName](#) property and the [Row](#) property.
2. Specify the list item with the [ColList](#) property.

Removing List Items

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

You can remove items (in a single-column control) or rows (in a multiple-column control) from the list in an fpCombo or fpList control in either of the following ways:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Use the [Action](#) property.

You must use this property if the control has more than 32,768 items.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Use the RemoveItem method.

For information on the RemoveItem method, refer to the Visual Basic documentation.

[Print](#)

[Copy](#)

[Close](#)

To remove list items using the Action property

Code

1. If you want to remove all items from the list, set the [Action](#) property to 3 (Clear).
2. If you want to delete a specific item or row of items,
 - a. Set the [Row](#) property to specify the row to remove.
 - b. Set the Action property to 4 (Delete Row).

Accessing List Items

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To select an item

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To return the index of the selected item

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To return the value of a specific item

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To return the value of the selected item

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

After the fpCombo or fpList control has been filled, you can work with the items in the list using various run-time properties. These properties let you

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Select an item at run time.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Return the index or value of a specific list item.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Return the value of the selected item.

The index reflects the position of list items. The index of the first item is zero.

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

With the fpList control, you can allow the user to select multiple items by setting the [MultiSelect](#) property to a value other than 0 (Single Only). The [SelCount](#) property returns the number of items selected, and the [SelMax](#) property sets a limit for the number of items the user can select.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

In an fpList control, when multiple items are selected, the index or value of the item that has the focus is returned.

[Print](#)

[Copy](#)

[Close](#)

To select an item

Code

1. If you are selecting an item in an fpCombo control, set the [ListIndex](#) property to the index of the item you want to select.
2. If you are selecting an item in an fpList control, you can perform either of the following actions:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Set the ListIndex property to the index of the item you want to select.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Set the [Row](#) property to the row number of the item and then set the [Selected](#) property to True.

[Print](#)

[Copy](#)

[Close](#)

To return the index of the selected item

Code

Return the index of the currently selected item with the [ListIndex](#) property.

Note The [TopIndex](#) property returns the index of the item currently displayed at the top of the list.

[Print](#)

[Copy](#)

[Close](#)

To return the value of a specific item

Designer Page

1. If you want to return the value of an item in a single-column list,
 - a. Specify the row. On the [Add Data designer page](#), either type the row number in the Row box or click the row in the preview area.
 - b. Select the [List](#) property in the properties list box.
The value is displayed in the box under List in the property value area.
2. If you want to return the value of a cell in a multiple-column control,
 - a. Specify the cell. On the Add Data designer page, either click the cell in the preview area or perform the following actions:
 - i. Select the column from the Col, Col Name, or Col ID drop-down list box.
 - ii. Type the row number in the Row box.
 - b. Select the [Collist](#) property in the properties list box.
The value is displayed in the box under Collist in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To return the value of a specific item

Code

1. If you want to return the value of an item in a single-column list,
 - a. Set the value of the [Row](#) property.
 - b. Set the value of a variable to the value of the [List](#) property.

Note The List property in List Pro is not an array property as it is in Visual Basic.

2. If you want to return the value of a cell in a multiple-column list,
 - a. Specify the column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.
 - b. Specify the row with the Row property.
 - c. Read the value of the [ColList](#) property.

[Print](#)

[Copy](#)

[Close](#)

To return the value of the selected item

Code

1. If you want to return the value of the selected item in a single-column list, set the value of a variable to the value of the [Text](#) property.

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

You do not have to set the [Row](#) property. The Text property automatically retrieves the row value for the selected item.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

For the fpCombo control, when the [Style](#) property is set to 0 (Drop-Down Combo) or 1 (Simple Combo), the Text property returns the text contained in the edit (or static) field. When the Style property is set to 2 (Drop-Down List), the Text property returns the value of the selected item.

2. If you want to return the value of a cell in the selected row in a multiple-column list,
 - a. Specify the column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.

Note You do not have to set the Row property. The [ColText](#) property automatically retrieves the row value for the selected row.

- b. Read the value of the ColText property.

Sorting List Items

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To sort a single-column list

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To sort a multiple-column list

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

While a List Pro control is being filled or after a List Pro control has been filled, you can sort list items to arrange them in alphabetic or numeric order. You can sort single- or multiple-column fpCombo or fpList controls. In multiple-column controls you can specify the sequence in which columns are sorted and the data type to help sort faster.

When you sort list items, the fpCombo or fpList control updates its index numbers. Adding an item to a sorted list might disrupt the sort order if you do not resort the items, depending upon where you add it.

You can temporarily suspend sorting by setting the [SortState](#) property. When you add numerous items to a list, for example, 10,000 records from a database, the fpCombo or fpList control sorts items after each addition. Setting the SortState property to 2 (Suspend) prevents the control from sorting items until the SortState property is set to 0 (Active) or 1 (Active (Re-Sort)). Temporarily suspending sorting at run time saves a significant amount of time if you are adding numerous items.

[Print](#)

[Copy](#)

[Close](#)

To sort a single-column list

Designer Page

1. Specify how you want to sort the list items.
 - a. On the [Sort designer page](#), select the [Sorted](#) property in the properties list box.
 - b. Select either the 1 - Ascending or 2 - Descending option button in the property value area.
2. Add items to your list, if needed.

For more information, see [Adding List Items](#).

[Print](#)

[Copy](#)

[Close](#)

To sort a single-column list

Code

1. If you are adding numerous items to your list, set the [SortState](#) property to 2 (Suspend) to suspend sorting until after you have added the items.
2. Set the [Sorted](#) property to 1 (Ascending) or 2 (Descending).
3. Add items to your list, if needed.

For more information, see [Adding List Items](#).

4. If you suspended sorting in step 1, set the SortState property to 1 (Active (Re-Sort)) to re-sort your items.

[Print](#)

[Copy](#)

[Close](#)

To sort a multiple-column list

Designer Page

1. Specify the column you want to sort. On the [Sort designer page](#), either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.
2. Specify how you want to sort the list items.
 - a. Select the [ColSorted](#) property from the properties list box.
 - b. Select either the 1 - Ascending or 2 - Descending option button under ColSorted in the property value area.
3. If you know the type of data in a particular column and want to improve sorting,
 - a. Select the [ColSortDataType](#) property in the properties list box.
 - b. Select the appropriate option button under ColSortDataType in the property value area.
4. Specify the sort sequence.
 - a. Select the [ColSortSeq](#) property from the properties list box.
 - b. If you are performing a sort on one column or if you are performing a sort on more than one column and this is the first column you want to sort by, type 0 in the box under ColSortSeq in the property value area. Otherwise, if you are performing a sort on more than one column, type the appropriate sequence number in the box under ColSortSeq in the property value area.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

5. Repeat steps 1-4 for each additional column.

[Print](#)

[Copy](#)

[Close](#)

To sort a multiple-column list

Code

1. If you are adding numerous items to your list, set the [SortState](#) property to 2 (Suspend) to suspend sorting until after you have added the items.
2. Specify the column you want to sort with the [Col](#), [ColFromID](#), or [ColFromName](#) property.
3. Set the [ColSorted](#) property to 1 (Ascending) or 2 (Descending).
4. If you know the type of data in a particular column and want to improve sorting, set the [ColSortDataType](#) property.
5. If you are performing a sort on one column or if you are performing a sort on more than one column and this is the first column you want to sort by, set the [ColSortSeq](#) property to 0. Otherwise, if you are performing a sort on more than one column, set the [ColSortSeq](#) property to the appropriate sequence number.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

6. Repeat steps 2-5 for each additional column.

For example, the following statements specify that the list is sorted first by the values in the third column and then by the values in the second column:

```
' second column
fpList1.Col = 1
' sort in ascending order
fpList1.ColSorted = 1
fpList1.ColSortDataType = LC_COLSORTDATATYPE_FLOAT
' sort second
fpList1.ColSortSeq = 2
' third column
fpList1.Col = 2
' sort in descending order
fpList1.ColSorted = 2
fpList1.ColSortDataType = LC_COLSORTDATATYPE_INTEGER
' sort first
fpList1.ColSortSeq = 1
```

7. If you previously suspended sorting, set the [SortState](#) property to 1 (Active (Re-Sort)) to re-sort your items.

Searching for List Items

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To search for list items by typing characters

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

To search for list items by supplying a search string

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can search for a specific list item in an fpCombo or fpList control. By default, only the first column is searched. For multiple-column fpCombo or fpList controls, you can specify which column to search.

The search method for an fpCombo control depends on whether it operates as a simple combo box, drop-down combo box, or drop-down list box. For more information on combo box styles, see [Combo Box Styles](#).

There are two search mechanisms:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

By typing a character or characters, for a list box or a drop-down list box, you can choose whether to search based on a single character or multiple characters. The control scrolls to the first item that begins with that character or characters.

For drop-down and simple combo boxes, no choice is necessary. Drop-down and simple combo boxes always perform a multiple-character search.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

By supplying a search string, you can perform the search at run time.

[Print](#)

[Copy](#)

[Close](#)

To search for list items by typing characters

Designer Page

1. If you are searching a multiple-column fpCombo or fpList control, designate which column to search.
 - a. On the [Search designer page](#), select the [ColumnSearch](#) property from the properties list box.
 - b. Type the number of the column in the box under ColumnSearch in the property value area.
2. For a drop-down list box or a list box, specify the type of search.
 - a. On the Search designer page, select the [AutoSearch](#) property from the properties list box.
 - b. Select any option button other than 0 - None under AutoSearch in the property value area.
3. At run time,
 - a. If you want to scroll the list to the first matching list item, set the fpCombo or fpList controls [TopIndex](#) property to the value of the [SearchIndex](#) property.

If you want to scroll to the first matching list item and select it, set the [ListIndex](#) property to the value of the SearchIndex property.
 - b. Set the [Action](#) property to 0 (Search) to perform the search.

[Print](#)

[Copy](#)

[Close](#)

To search for list items by typing characters

Browser/Code

1. If you are searching a multiple-column fpCombo or fpList control, set the [ColumnSearch](#) property to designate which column to search.
2. For a drop-down list box or a list box, set the [AutoSearch](#) property to a value other than 0 (None).
3. At run time,
 - a. If you want to scroll the list to the first matching list item, set the fpCombo or fpList controls [TopIndex](#) property to the value of the [SearchIndex](#) property.
If you want to scroll to the first matching list item and select it, set the [ListIndex](#) property to the value of the SearchIndex property.
 - b. Set the [Action](#) property to 0 (Search) to perform the search.

[Print](#)

[Copy](#)

[Close](#)

To search for list items by supplying a search string

Designer Page

1. If you want to consider case when searching,
 - a. On the [Search designer page](#), select the [SearchIgnoreCase](#) property from the property list box.
 - b. Select the False option button under SearchIgnoreCase in the property value area.
2. If you are searching a multiple-column fpCombo or fpList control, designate which column to search.
 - a. On the Search designer page, select the [ColumnSearch](#) property from the properties list box.
 - b. Type the number of the column in the box under ColumnSearch in the property value area.
3. Specify whether the search must find an exact match.
 - a. Select the [SearchMethod](#) property from the properties list box.
 - b. Select the appropriate option button to specify whether the search must find an exact match under SearchMethod in the property value area.
4. Specify the search text.
 - a. Select the [SearchText](#) property from the properties list box.
 - b. Type the search string in the box under SearchText in the property value area.
5. At run time,
 - a. Set the [Action](#) property to 0 (Search) to perform the search.
 - b. If you want to scroll the list to the first matching list item, set the fpCombo or fpList controls [TopIndex](#) property to the value of the [SearchIndex](#) property.
If you want to scroll to the first matching list item and select it, set the [ListIndex](#) property to the value of the SearchIndex property.

[Print](#)

[Copy](#)

[Close](#)

To search for list items by supplying a search string

Browser/Code

1. If you want to consider case when searching, set the [SearchIgnoreCase](#) property to False.
2. If you are using a multiple-column fpCombo or fpList control, designate which column to search with the [ColumnSearch](#) property.
3. Specify whether the search must find an exact match with the [SearchMethod](#) property.
4. At run time,
 - a. Specify the search string with the [SearchText](#) property.
 - b. Set the [Action](#) property to 0 (Search) to perform the search.
 - c. If you want to scroll the list to the first matching list item, set the fpCombo or fpList controls [TopIndex](#) property to the value of the [SearchIndex](#) property.

If you want to scroll to the first matching list item and select it, set the [ListIndex](#) property to the value of the SearchIndex property.

Customizing the Appearance of Selected Items

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can customize the way the control displays selected items in the list. You can specify the background color; pictures and picture alignment; text color, three-dimensional appearance, and alignment; and appearance of lines in the control.

[Print](#)

[Copy](#)

[Close](#)

To customize the appearance of selected items

Designer Page

1. On the [List](#) or [Line subtab of the ApplyTo designer page](#), from the List Apply To drop-down list box, select 4 - Sel Rows.
2. Follow the instructions in the appropriate topic listed in the following table.

For more information on . . .

Text color, appearance, and alignment
Pictures and picture alignment
Lines
Background color

See . . .

[Working with Text and Graphics](#)
Working with Text and Graphics
[Customizing Lines](#)
[Changing the Background Color](#)

[Print](#)

[Copy](#)

[Close](#)

To customize the appearance of selected items

Browser/Code

1. Set the [ListApplyTo](#) property to 4 (Sel Rows).
2. Follow the instructions in the appropriate topic listed in the following table.

For more information on . . .

Text color, appearance, and alignment
Pictures and picture alignment
Lines
Background color

See . . .

[Working with Text and Graphics](#)
Working with Text and Graphics
[Customizing Lines](#)
[Changing the Background Color](#)

Hiding the Focus Rectangle Around Selected Items

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

By default, a focus rectangle is drawn around the selected item. You can choose to have the control not display the focus rectangle.

Tip If you choose to display a three-dimensional appearance for list items, the list item might be more distinct if the focus rectangle is not displayed.

[Print](#)

[Copy](#)

[Close](#)

To not display the focus rectangle around selected items

Designer Page

1. On the [Miscellaneous designer page](#), select the [SelDrawFocusRect](#) property from the property list box.
2. Select the False option button under SelDrawFocusRect in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To not display the focus rectangle around selected items

Browser/Code

Set the [SelDrawFocusRect](#) property to False.

Wrapping List Items in a Single-Column Control

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

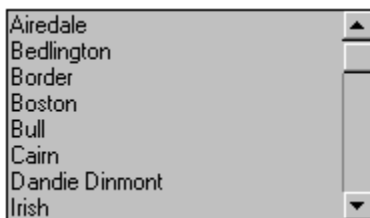
[Browser/Code Instructions](#)

Overview

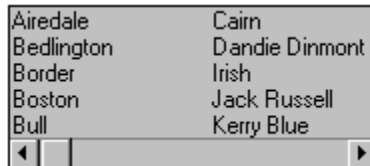
In a single-column fpCombo or fpList control, you can arrange list items in wrapping columns, filling the first column, then the second, and so on. The number of columns depends on the number of items, the width of the control, and the width of each wrapping column. The wrapped list cannot extend beyond the height of the control. The control displays vertical scroll bars to extend the data beyond the width of the control.

Caution When you set the [WrapList](#) property to True to arrange list items in wrapping columns, all multiple-column support is disabled and the designated-column property settings are ignored.

The following examples show the same list. This first list box shows the default single-column list.



This second list box shows the same list with three wrapping columns.



[Print](#)

[Copy](#)

[Close](#)

To wrap list items in a single-column control

Designer Page

1. Define the width of the wrapped columns.
 - a. On the [Misc subtab of the Appearance designer page](#), select the [WrapWidth](#) property from the properties list box.
 - b. Type the width of the wrapping columns in the box under WrapWidth in the property value area.
If you do not specify a width, the default column width applies.
2. Specify that you want the list to wrap in columns.
 - a. Select the [WrapList](#) property from the properties list box.
 - b. Select the True option button under WrapList in the property value area.
When you set the WrapList property to True, all multiple-column support is disabled and the designated-column property settings are ignored.

[Print](#)

[Copy](#)

[Close](#)

To wrap list items in a single-column control

Browser/Code

1. Specify the width of the wrapping columns with the [WrapWidth](#) property.

If you do not specify a width, the default column width applies.

2. Set the [WrapList](#) property to True.

When you set the WrapList property to True, all multiple-column support is disabled and the designated-column property settings are ignored.

Working with Databases

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

These instructions are for Visual Basic users only. If you want to perform data binding in a different development environment, please consult the documentation for that environment.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The list portions of fpCombo and fpList controls are designed for viewing data. They are read-only and cannot change values in the database table. However, information in the edit field of the fpCombo control can change values in the database table.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Binding a Control to a Database](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Binding Columns to Fields in a Database](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Adding Data to Nonbound Columns](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Binding to a Data Control on a Different Form](#)

Binding a Control to a Database

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser Instructions](#)

Overview

Using Visual Basic, you can bind ActiveX and VBX List Pro controls to databases and display the database records.

When you bind a List Pro control to a database using the default settings, the control displays all the available fields. The control automatically determines the number of columns to display and displays the fields in columns. The field headers are displayed as the column headers. In the fpCombo control, the edit field displays the fields run together with a column separator character between fields.

This topic describes how to bind a List Pro control to a database (or databases) and display either all fields in the list or one field in the list. [Binding Columns to Fields in a Database](#), describes how to bind specific columns in multiple-column controls to specific fields in the database table.

You can bind the edit field and the list in a combo box to different databases and different fields. In addition, you can synchronize the Data control and the selected item in the List Pro control.

You can customize a multiple column fpCombo or fpList control to display specific fields. For more information about creating multiple-column fpCombo and fpList controls, see [Creating Multiple Columns](#). For more information about binding columns to specific fields, see [Binding Columns to Fields in a Database](#).

The List Pro controls can be bound to any field type except binary. For more information, see [Data Binding](#).

[Print](#)

[Copy](#)

[Close](#)

To bind an fpCombo or fpList control to a database

Browser(VB only)

1. Create one or more Data controls on your form.
2. Bind the Data controls to databases with the DatabaseName property.
3. Bind the Data controls to specific database tables with the RecordSource property.
4. Create an fpCombo or fpList control on your form.

Note If you are binding your fpCombo or fpList control to a Data control on a different form, steps 5 and 6 are different. For instructions, see [Binding to a Data Control on a Different Form](#).

5. Set the DataSource property to the name of a Data control.

This step binds the database to the edit field of the combo box.

6. To bind the combo box list to a Data control, set the [DataSourceList](#) property to the name of a Data control.
7. If the database contains more than one data field and you want to designate one field for the control (creating a single-column control),
 - a. For a list box, set the DataField property to determine which field is displayed in the list.
 - b. For a combo box,
 - i. Set the DataField property to determine which field in the database bound to the edit field receives data from the control.
 - ii. If you bound the combo box list to a Data control in step 6, set the [DataFieldList](#) property to determine which field in the database bound to the list field receives data from the control.

If you do not set the DataField or DataFieldList property, the fpCombo or fpList control is bound to all the fields for each record.

8. Set the method for synchronizing the Data control and the selected item in the fpCombo or fpList control with the [DataSync](#) property.

Binding Columns to Fields in a Database

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser Instructions](#)

Overview

This topic describes how to bind specific columns in multiple-column controls to specific fields in the database table.

With a multiple-column fpCombo or fpList control, you can bind each column to a separate database field. The resulting list shows one record per row, with each column displaying the value stored in the field to which it is bound. You do not have to display all fields in the database.

By default, columns are automatically sized based on the length of the largest string in the corresponding bound database field. You can change how this occurs.

You can add data to bound fpCombo and fpList controls that is displayed in columns that are not bound to the database. For more information, see [Adding Data to Nonbound Columns](#).

[Print](#)

[Copy](#)

[Close](#)

To bind columns to separate fields


Designer Page

1. Create and bind the Data and List Pro controls.
 - a. Using the property browser, create one or more Data controls on your form.
 - b. Bind the Data controls to databases with the DatabaseName property.
 - c. Bind the Data controls to specific database tables with the RecordSource property.
 - d. Create an fpCombo or fpList control on your form.

Note If you are binding your fpCombo or fpList control to a Data control on a different form, steps 1.e and 1.f are different. For instructions, see [Binding to a Data Control on a Different Form](#).

 - e. Set the DataSource property to the name of a Data control.
This step binds the database to the edit field of the combo box.
 - f. To bind the combo box list to a Data control, set the [DataSourceList](#) property to the name of a Data control.
2. Create a multiple-column control.
 - a. On the [Specific subtab of the Columns designer page](#), select the [Columns](#) property from the properties list box.
 - b. Type the number of columns in the box under Columns in the property data area.
3. Set the method for synchronizing the Data control and the selected item.
 - a. On the [General subtab of the Data Binding designer page](#), select the [DataSync](#) property from the properties list box.
 - b. Select the appropriate option button under DataSync in the property value area.
4. Assign each column a field.
 - a. On the [Column subtab of the Data Binding designer page](#), either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area to specify the column to be bound.
 - b. Select the [ColDataField](#) property from the properties list box.
 - c. Type the database field name or number whose values you want to display in the column in the box under ColDataField in the property value area.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

- d. Repeat steps 4.a  4.c for all columns to be bound.
5. For a combo box, specify the fields to be bound to the edit field and determine which columns value from the list is written to the database.
 - a. On the Column subtab of the Data Binding designer page, select the [ColumnEdit](#) property in the properties list box.
 - b. Type the number of the column that you want to appear in the edit field in the box under ColumnEdit in the property value area.
 - c. Select the [ColumnBound](#) property in the properties value area.
 - d. Type the number of the column whose values are written to the database in the box under ColumnBound in the property value area.
 - e. In the property browser, set the DataField property to determine which field in the database bound to the edit field receives the value designated by the ColumnBound property.
 6. Customize the columns.

For example, create headers or change the column widths (after setting the [DataAutoSizeCols](#) property to 0 (Off)). For more information about customizing headers, see [Providing Column Headers](#). For more information about changing column width, see [Specifying the Column Width](#).

[Print](#)

[Copy](#)

[Close](#)

To bind columns to separate fields

Browser(VB only)

1. Create one or more Data controls on your form.
2. Bind the Data controls to databases with the DatabaseName property.
3. Bind the Data controls to specific database tables with the RecordSource property.
4. Create an fpCombo or fpList control on your form.

Note If you are binding your fpCombo or fpList control to a Data control on a different form, steps 5 and 6 are different. For instructions, see [Binding to a Data Control on a Different Form](#).

5. Set the DataSource property to the name of a Data control.
This step binds the database to the edit field of the combo box.
6. To bind the combo box list to a Data control, set the [DataSourceList](#) property to the name of a Data control.
7. Create a multiple-column control by specifying the number of columns with the [Columns](#) property.
8. Set the method for synchronizing the Data control and the selected item in the fpCombo or fpList control with the [DataSync](#) property.
9. At run time,
 - a. Specify the column to be bound with the [Col](#) property.
 - b. Designate the database field whose values you want to display in the column with the [ColDataField](#) property.
Note You must set the Col and ColDataField properties in an event procedure that occurs after the Columns property is set. You can set all these properties in the Form_Load procedure.
 - c. Repeat steps 9.a and 9.b for all columns to be bound.
10. For a combo box, specify the fields to be bound to the edit field and determine which columns value from the list is written to the database.
 - a. To display values in the edit field for a specific column in the list, set the [ColumnEdit](#) property.
 - b. To determine which columns value from the list is written to the database, set the [ColumnBound](#) property.
 - c. To determine which field in the database bound to the edit field receives the value designated by the ColumnBound property, set the DataField property.
11. Customize the columns.

For example, create headers with the [ColumnHeaderShow](#) property or change the column widths with the [ColWidth](#) property (after setting the [DataAutoSizeCols](#) property to 0 (Off)). For more information about customizing headers, see [Providing Column Headers](#). For more information about changing the column width, see [Specifying the Column Width](#).

Adding Data to Nonbound Columns

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser Instructions](#)

Overview

You can insert columns of nonbound data into an fpCombo or fpList control. That is, you can customize the control to display data along with the data displayed from a bound database.

[Print](#)

[Copy](#)

[Close](#)

To add data to nonbound columns in a bound fpCombo or fpList control

Designer Page

1. Create and bind the Data and List Pro controls.
 - a. Using the property browser, create one or more Data controls on your form.
 - b. Bind the Data controls to databases with the DatabaseName property.
 - c. Bind the Data controls to specific database tables with the RecordSource property.
 - d. Create an fpCombo or fpList control on your form.
 - e. Set the DataSource property to the name of a Data control.
This step binds the database to the edit field of the combo box.
 - f. To bind the combo box list to a Data control, set the [DataSourceList](#) property to the name of a Data control.
2. Create a multiple-column control.
 - a. On the [Specific subtab of the Columns designer page](#), select the [Columns](#) property from the properties list box.
 - b. Type the number of columns in the box under Columns in the property data area.
3. Set the method for synchronizing the Data control and the selected item.
 - a. On the [General subtab of the Data Binding designer page](#), select the [DataSync](#) property from the properties list box.
 - b. Select the appropriate option button under DataSync in the property value area.
4. Assign each column a field.
 - a. On the [Column subtab of the Data Binding designer page](#), either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area to specify the column.
 - b. Select the [ColDataField](#) property from the properties list box.
 - c. Type the database field name or number in the box under ColDataField in the property value area.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

 - d. Repeat steps 4.a
 - 4.c for all columns to be bound.
5. Add data to nonbound columns.
 - a. At run time, in the DataLoaded event for the control, specify the column to contain the nonbound data with the [Col](#) property.
 - b. Fill in each row of the column by setting the [Row](#) and [ColList](#) properties for each data item.

[Print](#)

[Copy](#)

[Close](#)

To add data to nonbound columns in a bound fpCombo or fpList control

Browser (VB only)

1. Create one or more Data controls on your form.
2. Bind the Data control to a database with the DatabaseName property.
3. Bind the Data control to a specific database table with the RecordSource property.
4. Create an fpCombo or fpList control on your form.
5. Set the DataSource property to the name of the Data control.
This step binds the database to the edit field of a combo box.
6. To bind the combo box list to a Data control, set the [DataSourceList](#) property to the name of a Data control.
7. Create a multiple-column control by specifying the number of columns with the [Columns](#) property.
8. Set the method for synchronizing the Data control and the selected item in the fpCombo or fpList control with the [DataSync](#) property.
9. At run time,
 - a. Specify the column to be bound with the [Col](#) property.
 - b. Designate the database field whose values you want to display in the list with the [ColDataField](#) property.
Note You must set the Col and ColDataField properties in an event procedure that occurs after the Columns property is set. You can set all these properties in the Form_Load procedure.
 - c. Repeat steps 9.a and 9.b for all columns to be bound.
10. Add data to nonbound columns.
 - a. At run time, in the DataLoaded event for the control, specify the column to contain the nonbound data with the Col property.
 - b. Fill in each row of the column by setting the [Row](#) and [ColList](#) properties for each data item.

Binding to a Data Control on a Different Form

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser Instructions](#)

Overview

Note This procedure is valid for the VBX controls only.

You can bind an fpCombo or fpList control to a Data control on a different form by using the ListPro_GetControlhWnd function. For combo boxes, you can bind the edit field separately from the list, and you can bind each to a Data control on different forms.

[Print](#)

[Copy](#)

[Close](#)

To bind to a Data control on a different form

Browser (VBX only)

1. Include the LP.BAS file in your project.
2. Create a Data control on a form.
3. Bind the Data control to a database with the DatabaseName property.
4. Bind the Data control to a specific database table with the RecordSource property.
5. On a different form, create an fpCombo or fpList control.
6. At run time, define the [DataSourceWnd](#) property for the control as the value of the [ListPro_GetControlhWnd](#) function for the Data control on the other form.

This step binds the database to the edit field of the combo box.

The following line defines the data source for the fpList control as the Data1 control on Form2.

```
fpList1.DataSourceWnd = ListPro_GetControlhWnd(Form2.Data1)
```

7. To bind the list portion of a combo box to a Data control on a different form, define the [DataSourceWndList](#) property at run time as the value of the ListPro_GetControlhWnd function for the Data control on the other form.

The following line defines the data source for the list portion of a combo box as the Data1 control on Form2.

```
fpCombo1.DataSourceWndList = ListPro_GetControlhWnd(Form2.Data1)
```

8. If you have a single-column control, to designate the database fields for the control,
 - a. For a list box, set the DataField property to determine which field is displayed in the list.
 - b. For a combo box, set the DataField property to determine which field in the database bound to the edit field receives data from the control.

If you bound the combo box list to a Data control in step 7, set the [DataFieldList](#) property to determine which field in the database bound to the list field receives data from the control.

If you do not set the DataField or DataFieldList property, the fpCombo or fpList control is bound to all the fields for each record. For more information, see [Binding a Control to a Database](#).

9. If you have a multiple-column control, to designate the database fields for the control,
 - a. Specify the number of columns for the fpCombo or fpList control with the [Columns](#) property.
 - b. At run time,
 - i. Specify the column to be bound with the [Col](#) property.
 - ii. Designate the database field whose values you want to display in the column with the [ColDataField](#) property.

Note You must set the Col and ColDataField properties in an event procedure that occurs after the Columns property is set. You can set all these properties in the Form_Load procedure.

- iii. Repeat steps 9.b.i and 9.b.ii for all columns to be bound.

If you do not set the ColDataField property, the fpCombo or fpList control is bound to all the fields for each record. For more information, see [Binding Columns to Fields in a Database](#).

Using Virtual Mode

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

If the list has a large number of items (for example, if the control is bound to a large database), the control must read all items into memory before displaying the list. Virtual mode speeds up the performance of the fpCombo or fpList control because the control reads only the number of items necessary to fill the portion of the list that is displayed. For more information, see [Virtual Mode](#).

You can specify how many rows are read into memory and how many rows are read into the buffer area at a time. In addition, you can display a customized vertical scroll bar to use with virtual mode.

[Print](#)

[Copy](#)

[Close](#)

To use virtual mode

Designer Page

1. Turn on virtual mode.
 - a. On the [Virtual Mode designer page](#), select the [VirtualMode](#) property from the properties list box.
 - b. Select the True option button under VirtualMode in the property value area.
2. Specify the number of items to be read into memory (the size of the "page").
 - a. Select the [VirtualPageSize](#) property from the properties list box.
 - b. Type the number of items to be read into memory in the box under VirtualPageSize in the property value area.
3. Specify the number of virtual pages to be read into the buffer area at one time.
 - a. Select the [VirtualPagesAhead](#) property from the properties list box.
 - b. Type the number of virtual pages in the box under VirtualPagesAhead in the property value area.
4. If you want to use the default scroll bar,
 - a. Select the [VRowCount](#) property from the properties list box.
 - b. Type the total number of database records in the box under VRowCount in the property value area.

Providing the number of records enables the scroll box to reflect the position of the current record without reading ahead to the end of the table.

If you do not know the exact number of database records, type a value that approximates the number of records in the box under VRowCount in the property value area.
5. If you want to use a custom scroll bar,
 - a. Select the [VScrollSpecial](#) property in the properties list box.
 - b. Select the True option button under VScrollSpecial in the property value area.
 - c. If you want to remove some or all of the default scroll arrows,
 - i. Select the [VScrollSpecialType](#) property in the properties list box.
 - ii. Type the appropriate value in the box under VScrollSpecialType in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To use virtual mode

Browser/Code

1. Set the [VirtualMode](#) property to True to turn on virtual mode.
2. Specify the number of items to be read into memory (the size of the "page") with the [VirtualPageSize](#) property.
3. Specify the number of virtual pages to be read into the buffer area at one time with the [VirtualPagesAhead](#) property.
4. If you want to use the default scroll bar, set the [VRowCount](#) property to the total number of database records.

Providing the number of records enables the scroll box to reflect the position of the current record without reading ahead to the end of the table.

If you do not know the exact number of database records, set the VRowCount property to a value that approximates the number of records.

5. If you want to use a custom scroll bar,
 - a. Set the [VScrollSpecial](#) property to True.
 - b. If you want to remove some or all of the default scroll arrows, set the [VScrollSpecialType](#) property.

Using the FarPoint Property Designer

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Overview](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Starting the FarPoint Property Designer](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Using the FarPoint Property Designer Pages](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[FarPoint Property Designer Pages](#)

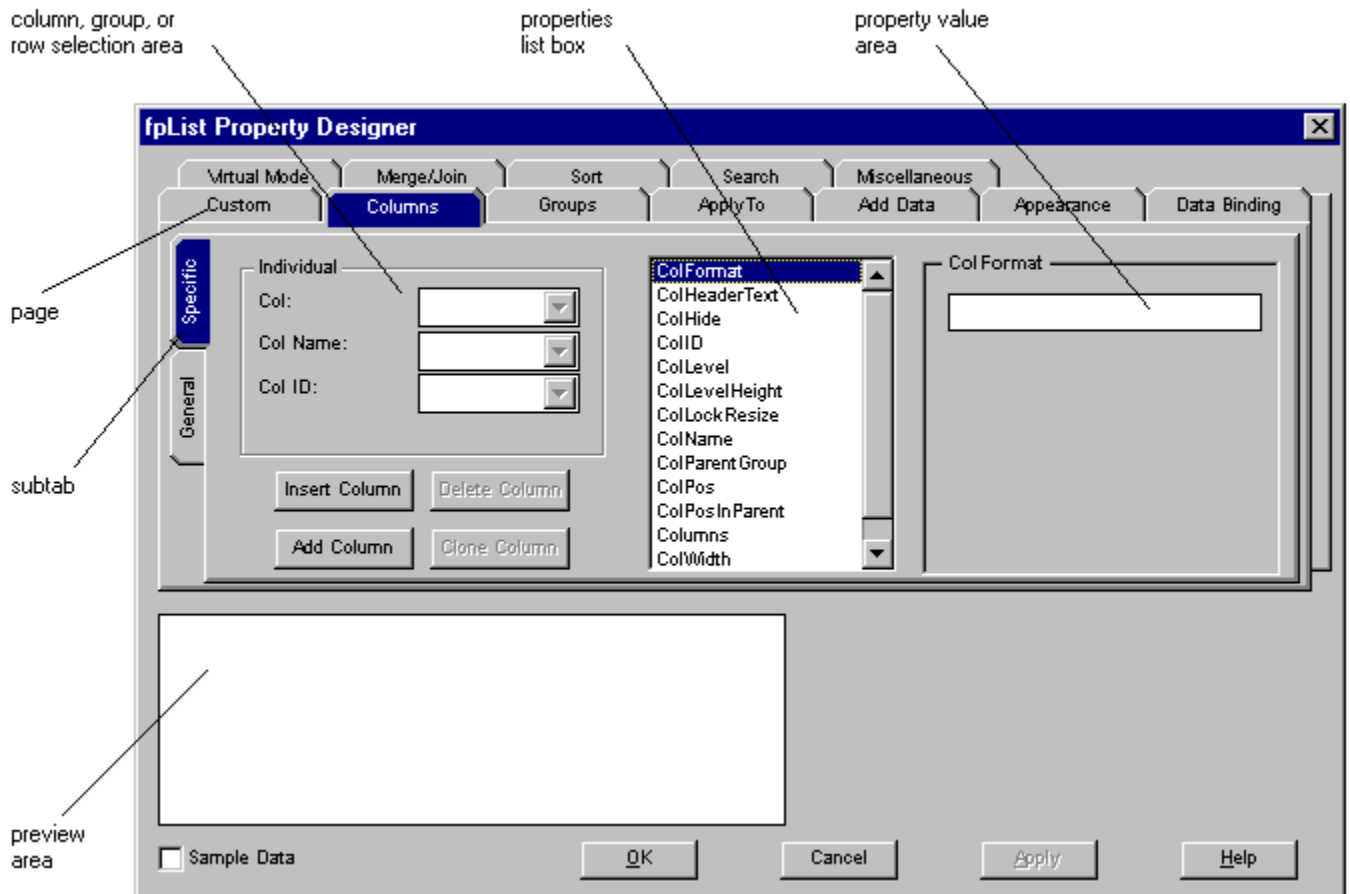
Overview

Note The FarPoint Property Designer can be used only with the List Pro ActiveX or VBX controls.

You can use the FarPoint Property Designer to design the look of the fpCombo or fpList control. You can set most design-time and some run-time properties using the designer. By setting properties at design time instead of run time, you can view the run-time appearance of the control and your application will run faster because less code is being executed during the Form_Load event.

Specific instructions for using the FarPoint Property Designer to create and customize List Pro controls are included in the [How-to Guides](#). Follow the instructions marked "Designer Page."

The FarPoint Property Designer pages are collections of tabs that present most design-time and some run-time properties for the List Pro controls. The FarPoint Property Designer pages are organized by control characteristics. Each page contains one or more properties that let you set characteristics of the List Pro controls.



The FarPoint Property Designer creates a "preview" of the List Pro control and displays it at the bottom of each page. You can populate the control with "sample" data or with "live" data. When you click the OK button, all changes made in the designer pages are applied to the selected List Pro control on your form or dialog.

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

data.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If you add data using the FarPoint Property Designer and you are displaying the sample data, you will not see the "live" data. If you add data and you are not displaying the sample data, you will see the "live" data.

You will not see the effect of most property changes unless the "preview" control is populated with

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Initially two rows of sample data are provided. Use the [ListCount](#) property on the Add Data designer page to add more sample data rows to the preview area.

Often, the designer page will have subtabs. As shown in the preceding figure, the [Columns](#) page has two subtabs, Specific and General.

If you are required to specify a cell, column, group, or row before setting a property, a box is provided for the choice or you can click the cell, column, group, or row in the preview area. When you click a cell, column, or group in the preview area, it will not be highlighted. However, the boxes in the column, row, or group selection area will reflect where you clicked. For example, if you click on the cell (2, 3), the Col box will show the value "2" and the Row box will show the value "3".

Each designer page contains standard buttons: OK, Cancel, Apply, and Help. Choose the OK button to close the designer page window and to apply any changes you have made. Choose the Cancel button to close the window without applying any changes. Choose the Apply button to apply your changes without closing the window. Choose the Help button to access context-sensitive help about the current designer page. Note that any time you change the active designer page, the changes you have made on that page are applied to the control.

Notes for ActiveX users

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The FarPoint Property Designer pages replace the ActiveX property pages.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Stock properties such as Enabled and BackColor are often provided on designer pages. Most design-time custom properties are provided as well.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Standard extender properties are not provided on FarPoint Property Designer pages because they are not necessarily supported by all containers.

Both the fpCombo and fpList controls provide 12 designer pages: Custom, Columns, Groups, ApplyTo, Add Data, Appearance, Data Binding, Virtual Mode, Merge/Join, Sort, Search, and Miscellaneous. For detailed information on each page, see [FarPoint Property Designer Pages](#).

Starting the FarPoint Property Designer

Before you use the FarPoint Property Designer, you should be familiar with the List Pro control features described in [List Pro Features](#).

To start the FarPoint Property Designer

1. If you are using the List Pro ActiveX control, display the pop-up menu by clicking the List Pro control with the right mouse button and then choose Property Designer.
2. If you are using the List Pro VBX control, place a List Pro control on the form or dialog and perform one of the following:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Double-click the fpPropertyDesigner property in the Properties window or dialog box.

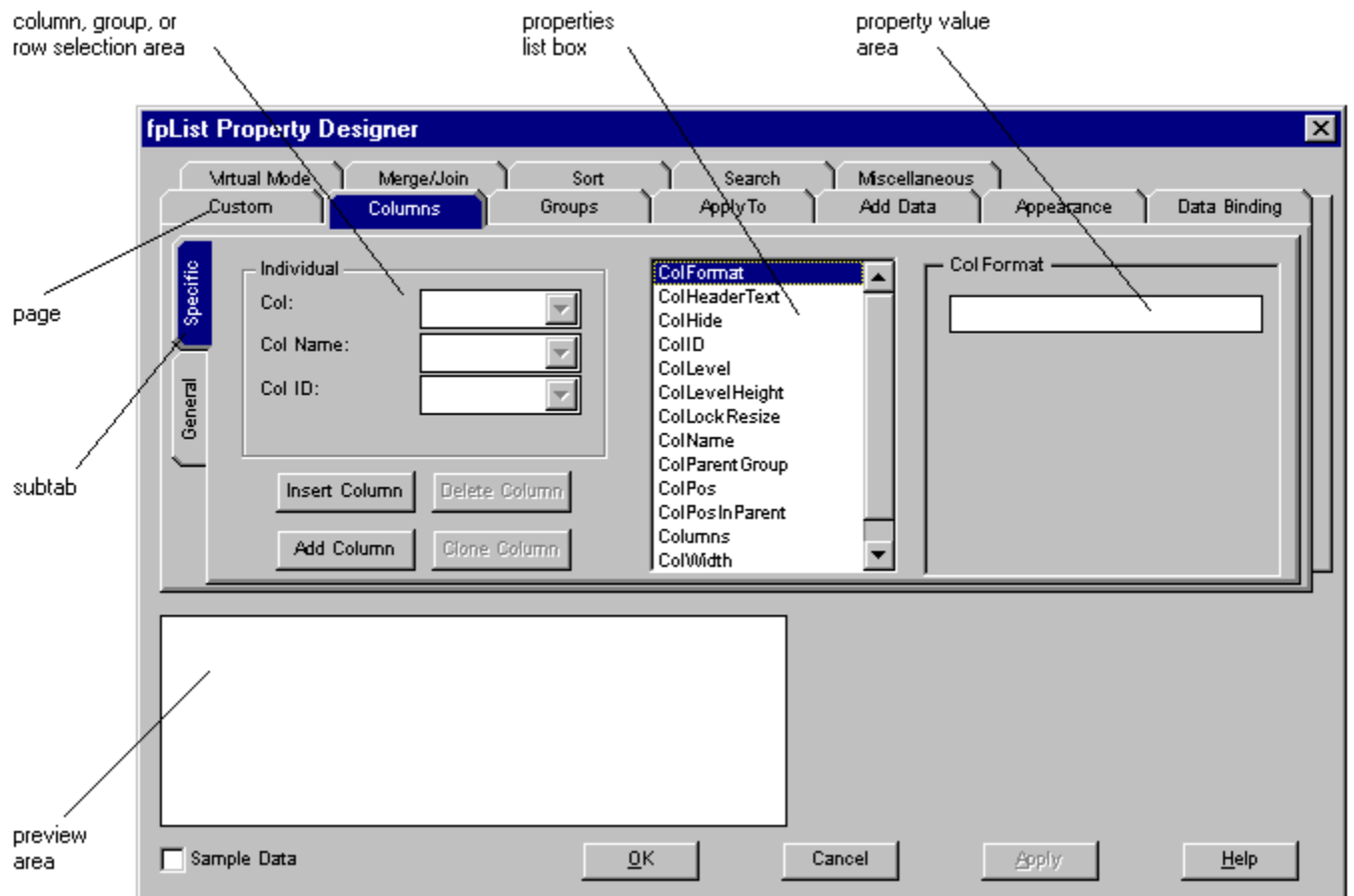
PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Select a List Pro control and press F4 twice.

The FarPoint Property Designer pages appear.

Using the FarPoint Property Designer Pages

Refer to the following figure for identification of specific areas of the designer page.



To use the FarPoint Property Designer pages

1. Select a tab.
2. Click a property in the properties list box.
3. If you chose the [ApplyTo designer page](#) and either the List or Line subtab, select the area to which the property applies from the Applies To drop-down list box.
4. If you need to set the column or group, select the column or group by number, name, or identification number.
You can also click the group or column in the preview area.
5. If you need to set the row, enter the number in the Row box.
You can also click the row in the preview area.
6. If you need to specify a cell, select the column by number, name, or identification number and enter the row number in the Row box.
You can also click the cell in the preview area.
7. Enter a value or select an option in the property value area.

FarPoint Property Designer Pages

The fpCombo and fpList controls provide 12 designer pages for customizing your interface.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Custom](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Columns](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Groups](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Apply To](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Add Data](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Appearance](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Data Binding](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Virtual Mode](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Merge/Join](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Sort](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Search](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Miscellaneous](#)

Custom Designer Page

The Custom designer page contains an alphabetical list of all properties that you can set using the FarPoint Property Designer. You set properties on this page much like you set properties with the property browser.

The following items are available on the [Custom designer page](#):



The designer page referenced in the previous paragraph is unique to the fpList control. If you want to view the Custom designer page for the fpCombo control, [click here](#).

Custom Designer Page

Item on designer page	Description	Corresponding property
Properties list box	N/A	Most FarPoint custom properties
Property value	Displays option buttons, boxes, color charts, or preview areas depending on the property chosen	N/A

Columns Designer Page

The Columns designer page lets you set most column-related properties. The Columns designer page contains two subtabs, Specific and General.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Specific](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[General](#)

Columns Designer Page—Specific Subtab

The following items are available on the Columns designer page and the [Specific subtab](#):

Columns Designer Page—Specific Subtab

Item on designer page	Description	Corresponding property
Properties list box	Displays properties needed to set up columns in the List Pro control	ColFormat ColHeaderText ColHide ColID ColLevel ColLevelHeight ColLockResize ColName ColParentGroup ColPos ColPosInParent Columns ColWidth
Property value	Displays option buttons and boxes depending on the property chosen	N/A
Insert Column button	Inserts a column to the left of the currently selected column	Action - Insert Column
Delete Column button	Deletes the currently selected column	Action - Delete Column
Add Column button	Adds a column to the right of the last column	Action - Add Column
Clone Column button	Copies the property settings of the currently selected column and inserts the clone column to the right of the currently selected column Note: No data is copied.	Action - Clone Column
Individual Col drop-down list box	Specifies the column number	Col
Individual Col Name drop-down list box	Specifies the column by name	ColFromName
Individual Col ID drop-down list box	Specifies the column by identification number	ColFromID

Columns Designer Page—General Subtab

The following items are available on the Columns designer page and the [General subtab](#):



The subtab referenced in the previous paragraph is unique to the fpList control. If you want to view the General subtab for the fpCombo control, [click here](#).

Columns Designer Page—General Subtab

Item on designer page	Description	Corresponding property
Properties list box	Displays miscellaneous column properties	AllowColDragDrop AllowColResize ColsFrozen ColumnEdit (fpCombo only) ColumnHeaderHeight ColumnHeaderShow ColumnLevels ColumnSeparatorChar ColumnWidthScale
Property value	Displays option buttons or boxes depending on the property chosen	N/A

Groups Designer Page

The Groups designer page lets you set all group-related properties. The Groups designer page contains two subtabs, Specific

and General.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Specific

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

General

Groups Designer Page—Specific Subtab

The following items are available on the Groups designer page and the [Specific subtab](#):

Groups Designer Page—Specific Subtab

Item on designer page	Description	Corresponding property
Properties list box	Displays properties needed to set up groups in the List Pro control	Groups GrpHeaderText GrpHide GrpID GrpLockResize GrpName GrpParentGroup GrpPos GrpPosInParent GrpWidth
Property value	Displays option buttons or boxes depending on the property chosen	N/A
Insert Group button	Inserts a group to the right of the currently selected group	Action - Insert Group
Delete Group button	Deletes the currently selected group	Action - Delete Group
Add Group button	Adds a group to the right of the last column	Action - Add Group
Individual Grp drop-down list box	Specifies the group number	Grp
Individual Grp Name drop-down list box	Specifies the group by name	GrpFromName
Individual Grp ID drop-down list box	Specifies the group by identification number	GrpFromID

Groups Designer Page—General Subtab

The following items are available on the Groups designer page and the [General subtab](#):

Groups Designer Page—General Subtab

Item on designer page	Description	Corresponding property
Properties list box	Displays miscellaneous group properties	AllowGrpDragDrop AllowGrpResize GroupHeaderHeight GroupHeaderShow GrpsFrozen
Property value	Displays option buttons or boxes depending on the property chosen	N/A

ApplyTo Designer Page

The ApplyTo designer page lets you apply designated-list and designated-line properties to different parts of the control. The ApplyTo designer page contains two subtabs, List and Line.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[List](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Line](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ApplyTo Designer Page **List Subtab**

The following items are available on the ApplyTo designer page and the [List subtab](#):

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ApplyTo Designer Page **List Subtab**

Item on designer page	Description	Corresponding property
Properties list box	N/A	AlignH AlignV BackColor Font FontEmpty ForeColor List3DText List3DTextHighlightColor List3DTextOffset List3DTextShadowColor MultiLine Picture PictureAlignH PictureAlignV PictureSel TextOrientation

Property value	Displays option buttons, boxes, color charts, or preview areas depending on the property chosen	N/A
----------------	---	-----

List Apply To drop-down list box	Specifies the part of the control to which the property is applied	ListApplyTo
----------------------------------	--	-----------------------------

The following items are displayed as appropriate when List Apply To is set to 9 - Single Col Header, 10 - Single Group Header, 11 - Single Group, or 12 - Single Item:

Individual Col drop-down list box	Specifies the column number	Col
Individual Col Name drop-down list box	Specifies the column by name	ColFromName
Individual Col ID drop-down list box	Specifies the column by identification number	ColFromID
Individual Grp drop-down list box	Specifies the group number	Grp
Individual Grp Name drop-down list box	Specifies the group by name	GrpFromName
Individual Grp ID drop-down list box	Specifies the group by identification number	GrpFromID
Row box	Specifies the row	Row

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ApplyTo Designer Page **Line Subtab**

The following items are available on the ApplyTo designer page and the [Line subtab](#):

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ApplyTo Designer Page **Line Subtab**

Item on designer page	Description	Corresponding property
Properties list box	N/A	Line3DDark Line3DLight Line3DWidth LineColor LineStyle

Property value	Displays option buttons, boxes, or color charts, depending on the property chosen	LineWidth N/A
List Apply To drop-down list box	Specifies the part of the control to which the property is applied	ListApplyTo
Line Apply To drop-down list box	Specifies the line to which the property is applied	LineApplyTo
The following items are displayed as appropriate when List Apply To is set to 9 - Single Col Header, 10 - Single Group Header, 11 - Single Group, or 12 - Single Item:		
Individual Col drop-down list box	Specifies the column number	Col
Individual Col Name drop-down list box	Specifies the column by name	ColFromName
Individual Col ID drop-down list box	Specifies the column by identification number	ColFromID
Individual Grp drop-down list box	Specifies the group number	Grp
Individual Grp Name drop-down list box	Specifies the group by name	GrpFromName
Individual Grp ID drop-down list box	Specifies the group by identification number	GrpFromID
Row box	Specifies the row	Row

Add Data Designer Page

The Add Data designer page lets you add data to the List Pro control.

The following items are available on the [Add Data designer page](#):

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The designer page referenced in the previous paragraph is unique to the fpList control. If you want to view the Add Data designer page for the fpCombo control, [click here](#).

Add Data Designer Page

Item on designer page	Description	Corresponding property
Properties list box	N/A	ColList InsertRow List ListCount Text (fpCombo only)
Property value	Displays option buttons or boxes depending on the property chosen	N/A
The following items are displayed when necessary:		
Individual Col drop-down list box	Specifies the column number	Col
Individual Col Name drop-down list box	Specifies the column by name	ColFromName
Individual Col ID drop-down list box	Specifies the column by identification number	ColFromID
Row box	Specifies the row	Row
Delete Row button	Deletes the currently selected row	Action

Appearance Designer Page

The Appearance designer page lets you customize the appearance of the List Pro control. The Appearance designer page contains four subtabs, Border, Scroll, Color, and Misc.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Border](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Scroll](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Color](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Misc](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Appearance Designer Page Border Subtab

The following items are available on the Appearance designer page and the [Border subtab](#):

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The subtab referenced in the previous paragraph is unique to the fpList control. If you want to view the Border subtab for the fpCombo control, [click here](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Appearance Designer Page Border Subtab

Item on designer page	Description	Corresponding property
Properties list box	N/A	Appearance BorderDropShadow BorderDropShadowWidth BorderStyle BorderWidth ThreeDFrameWidth ThreeDInsideStyle ThreeDInsideWidth ThreeDOnFocusInvert ThreeDOutsideStyle ThreeDOutsideWidth
Property value	Displays option buttons or boxes depending on the property chosen	N/A

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Appearance Designer Page Scroll Subtab

The following items are available on the Appearance designer page and the [Scroll subtab](#):

Item on designer page	Description	Corresponding property
Properties list box	N/A	ScrollBarH ScrollBarV ScrollHInc ScrollHScale
Property value	Displays option buttons or boxes depending on the property chosen	N/A

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Appearance Designer Page Color Subtab

The following items are available on the Appearance designer page and the [Color subtab](#):

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The subtab referenced in the previous paragraph is unique to the fpList control. If you want to view the Color subtab for the fpCombo control, [click here](#).

Item on designer page	Description	Corresponding property
Properties list box	N/A	BorderColor BorderDropShadowColor

[BorderGrayAreaColor](#)
[GrayAreaColor](#) (fpCombo only)
[ListGrayAreaColor](#)
[ThreeDFrameColor](#)
[ThreeDInsideHighlightColor](#)
[ThreeDInsideShadowColor](#)
[ThreeDOutsideHighlightColor](#)
[ThreeDOutsideShadowColor](#)

Property value Displays color charts N/A

Appearance Designer Page

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Misc Subtab

The following items are available on the Appearance designer page and the [Misc subtab](#):

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The subtab referenced in the previous paragraph is unique to the fpList control. If you want to view the Miscellaneous subtab for the fpCombo control, [click here](#).

Appearance Designer Page

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Misc Subtab

Item on designer page	Description	Corresponding property
Properties list box	N/A	ComboGap (fpCombo only) EditHeight (fpCombo only) ExtendCol ExtendRow HighestPrecedence ListLeftOffset (fpCombo only) ListWidth (fpCombo only) MaxDrop (fpCombo only) MaxEditLen (fpCombo only) NoIntegralHeight RowHeight Style (fpCombo only) WrapList WrapWidth

Property value Displays option buttons or boxes depending on the property chosen N/A

The following item is displayed when necessary:

Row box Specifies the row [Row](#)

Data Binding Designer Page

The Data Binding designer page lets you set data binding properties. The Data Binding designer page contains two subtabs, Columns and General.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Columns](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[General](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Data Binding Designer Page

✓	✓	✓	✓	✓
---	---	---	---	---

Columns Subtab

The following items are available on the Data Binding designer page and the [Columns subtab](#):

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The subtab referenced in the previous paragraph is unique to the fpList control. If you want to view the Columns subtab for the fpCombo control, [click here](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Data Binding Designer Page

✓	✓	✓	✓	✓
---	---	---	---	---

Columns Subtab

Item on designer page	Description	Corresponding property
Properties list box	N/A	ColDataField ColFormat ColumnBound (fpCombo only) ColumnEdit (fpCombo only)
Property value	Displays option buttons or boxes depending on the property chosen	N/A
The following items are displayed when necessary:		
Individual Col drop-down list box	Specifies the column number	Col
Individual Col Name drop-down list box	Specifies the column by name	ColFromName
Individual Col ID drop-down list box	Specifies the column by identification number	ColFromID

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Data Binding Designer Page

✓	✓	✓	✓	✓
---	---	---	---	---

General Subtab

The following items are available on the Data Binding designer page and the [General subtab](#):

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The subtab referenced in the previous paragraph is unique to the fpList control. If you want to view the General subtab for the fpCombo control, [click here](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Data Binding Designer Page

✓	✓	✓	✓	✓
---	---	---	---	---

General Subtab

Item on designer page	Description	Corresponding property
Properties list box	N/A	DataAutoHeadings DataAutoSizeCols DataFieldList (fpCombo only) DataSync
Property value	Displays option buttons or boxes depending on the property chosen	N/A

Virtual Mode Designer Page

The Virtual Mode designer page lets you specify whether virtual mode is on or off and defines how the control looks during virtual mode.

The following items are available on the [Virtual Mode designer page](#):

Item on designer page	Description	Corresponding property
Properties list box	N/A	VirtualMode VirtualPagesAhead

[VirtualPageSize](#)
[VRowCount](#)
[VScrollSpecial](#)
[VScrollSpecialType](#)

Property value Displays option buttons or boxes depending on the property chosen N/A

Merge/Join Designer Page

The Merge/Join designer page lets you specify which columns or rows are merged and which cells are joined.

The following items are available on the [Merge/Join designer page](#):

Merge/Join Designer Page

Item on designer page	Description	Corresponding property
Properties list box	N/A	ColMerge JoinID MergeAdjustView RowMerge
Property value	Displays option buttons or boxes depending on the property chosen	N/A

The following items are enabled when necessary:

Individual Col drop-down list box	Specifies the column number	Col
Individual Col Name drop-down list box	Specifies the column by name	ColFromName
Individual Col ID drop-down list box	Specifies the column by identification number	ColFromID
Row box	Specifies the row number	Row

Sort Designer Page

The Sort designer page lets you specify the parameters necessary to perform a sort on the list items in the control.

The following items are available on the [Sort designer page](#):

Sort Designer Page

Item on designer page	Description	Corresponding property
Properties list box	N/A	ColSortDataType ColSorted ColSortSeq Sorted SortState
Property value	Displays option buttons or boxes depending on the property chosen	N/A

The following items are enabled when necessary:

Individual Col drop-down list box	Specifies the column number	Col
Individual Col Name drop-down list box	Specifies the column by name	ColFromName
Individual Col ID drop-down list box	Specifies the column by identification number	ColFromID

Search Designer Page

The Search designer page lets you specify the parameters necessary to perform a search of the list items in the control.

The following items are available on the [Search designer page](#):

Search Designer Page

Item on designer page	Description	Corresponding property
Properties list box	N/A	AutoSearch

[ColumnSearch](#)
[SearchIgnoreCase](#)
[SearchMethod](#)

Property value Displays option buttons or boxes,
depending on the property chosen N/A

Miscellaneous Designer Page

The Miscellaneous designer page lets you set properties that enable certain List Pro events, that make the control a read-only control, specify whether a focus rectangle is drawn around selected items, and specify the maximum number of items that can be selected in an fpList control.

The following items are available on the [Miscellaneous designer page](#):

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

The designer page referenced in the previous paragraph is unique to the fpList control. If you want to view the Miscellaneous designer page for the fpCombo control, [click here](#).

Miscellaneous Designer Page

Item on designer page	Description	Corresponding property
Properties list box	N/A	EnableKeyEvents EnableMouseEvents EnableTopChangeEvent MultiSelect (fpList only) ReadOnly SelDrawFocusRect SelMax (fpList only)

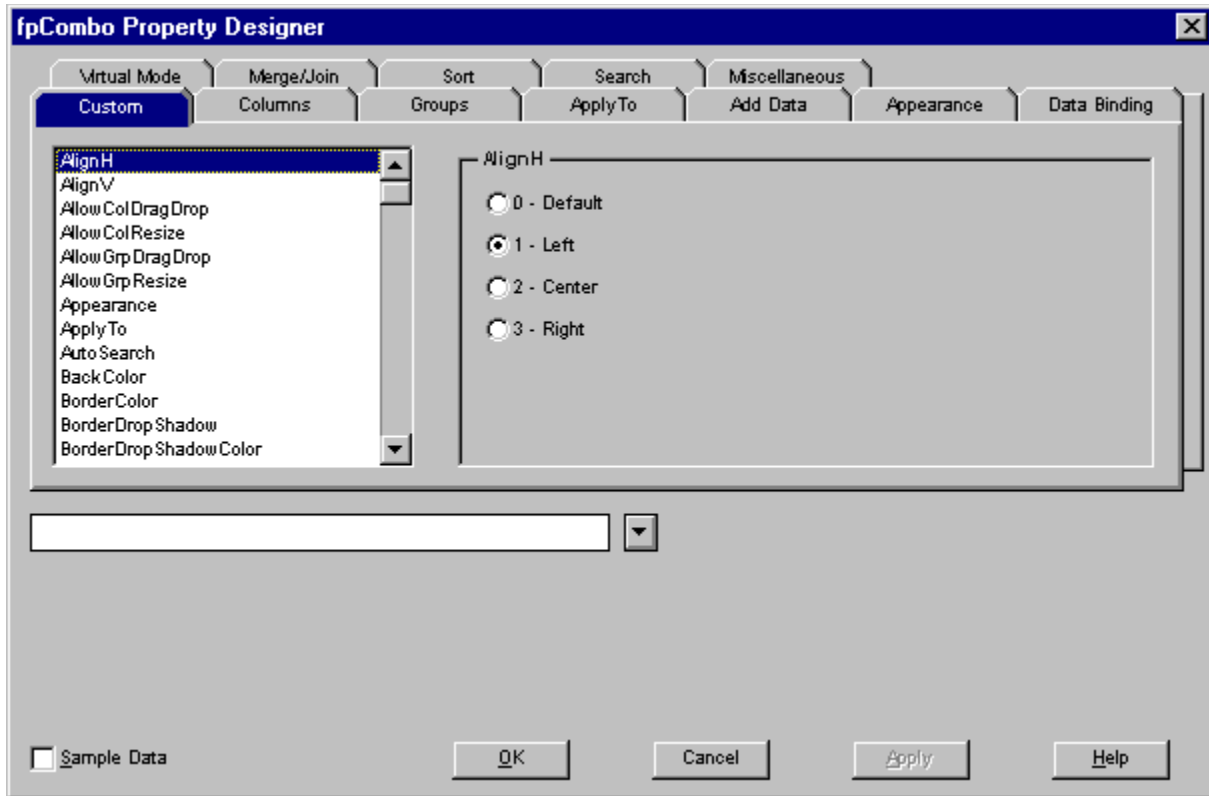
Property value Displays option buttons or boxes,
depending on the property chosen N/A

Custom

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

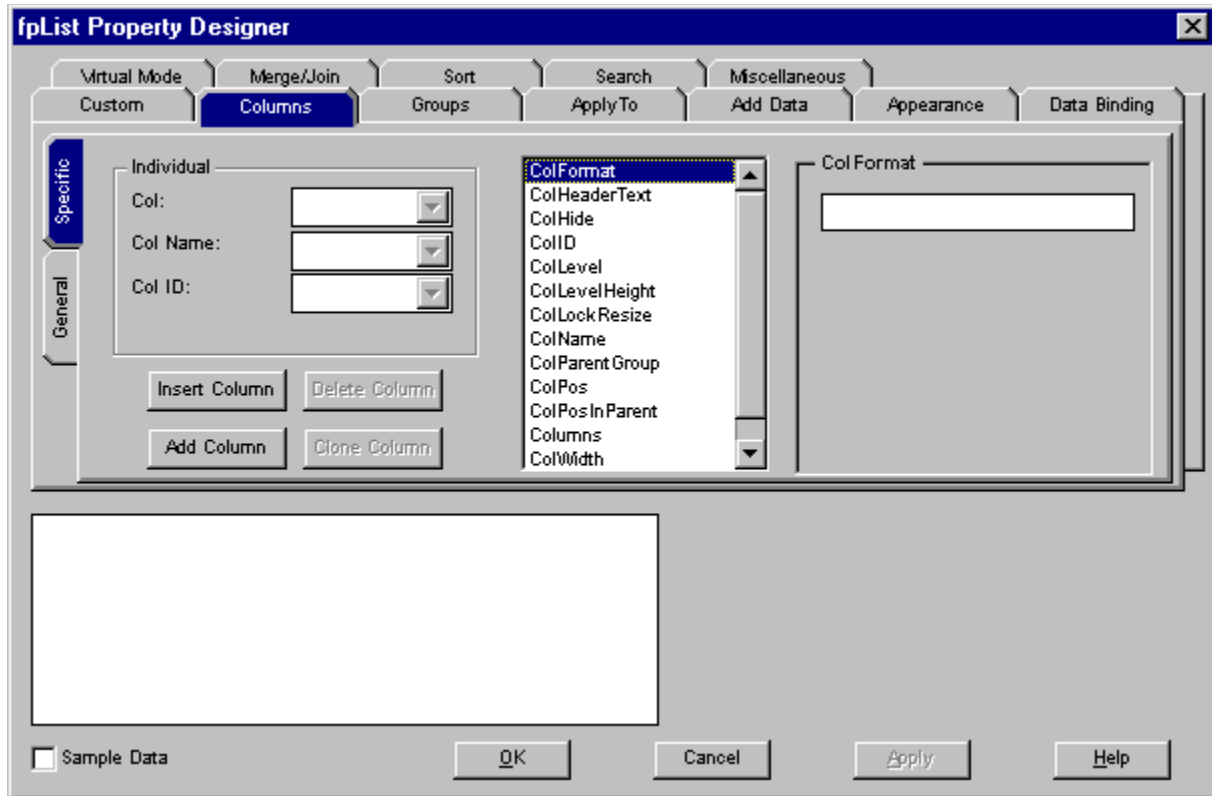
PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Custom fpCombo Control



PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Columns **Specific Subtab**



PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Columns **General Subtab**

fpList Property Designer [X]

Virtual Mode Merge/Join Sort Search Miscellaneous
 Custom **Columns** Groups Apply To Add Data Appearance Data Binding

Specific
 General

Allow Col Drag Drop
 Allow Col Resize
 Cols Frozen
 ColumnHeaderHeight
 ColumnHeaderShow
 ColumnLevels
 ColumnSeparatorChar
 ColumnWidth Scale

Allow Col Drag Drop
 0 - Off
 1 - All Cols
 2 - Non Frozen Cols

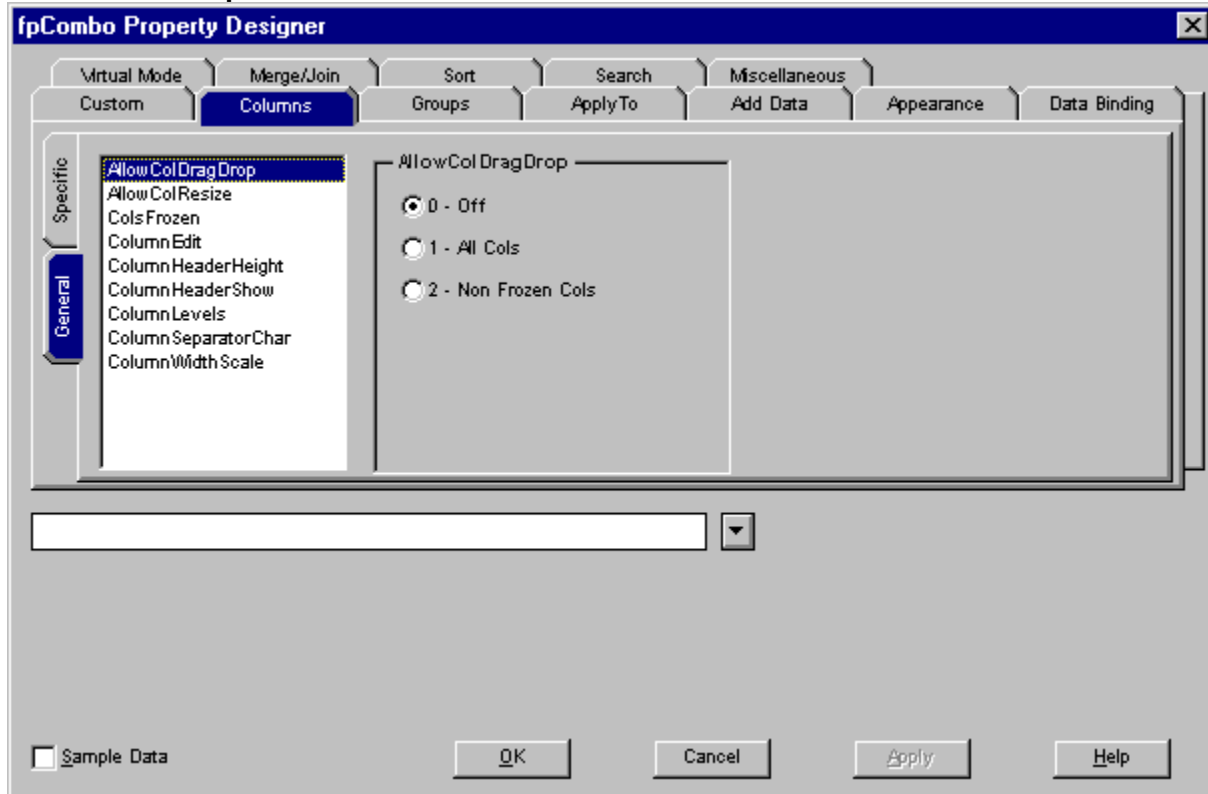
Sample Data

OK Cancel Apply Help

	PD	RD	WR	RT	DT
Columns	✓	✓	✓	✓	✓

General Subtab

	PD	RD	WR	RT	DT
fpCombo Control	✓	✓	✓	✓	✓



PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Groups **Specific Subtab**

fpList Property Designer [X]

Virtual Mode | Merge/Join | Sort | Search | Miscellaneous
 Custom | Columns | **Groups** | Apply To | Add Data | Appearance | Data Binding

Specific | General

Individual
 Grp: []
 Grp Name: []
 Grp ID: []

Insert Group | Delete Group
 Add Group

Groups
 GrpHeaderText
 GrpHide
 GrpID
 GrpLockResize
 GrpName
 GrpParentGroup
 GrpPos
 GrpPosInParent
 GrpWidth

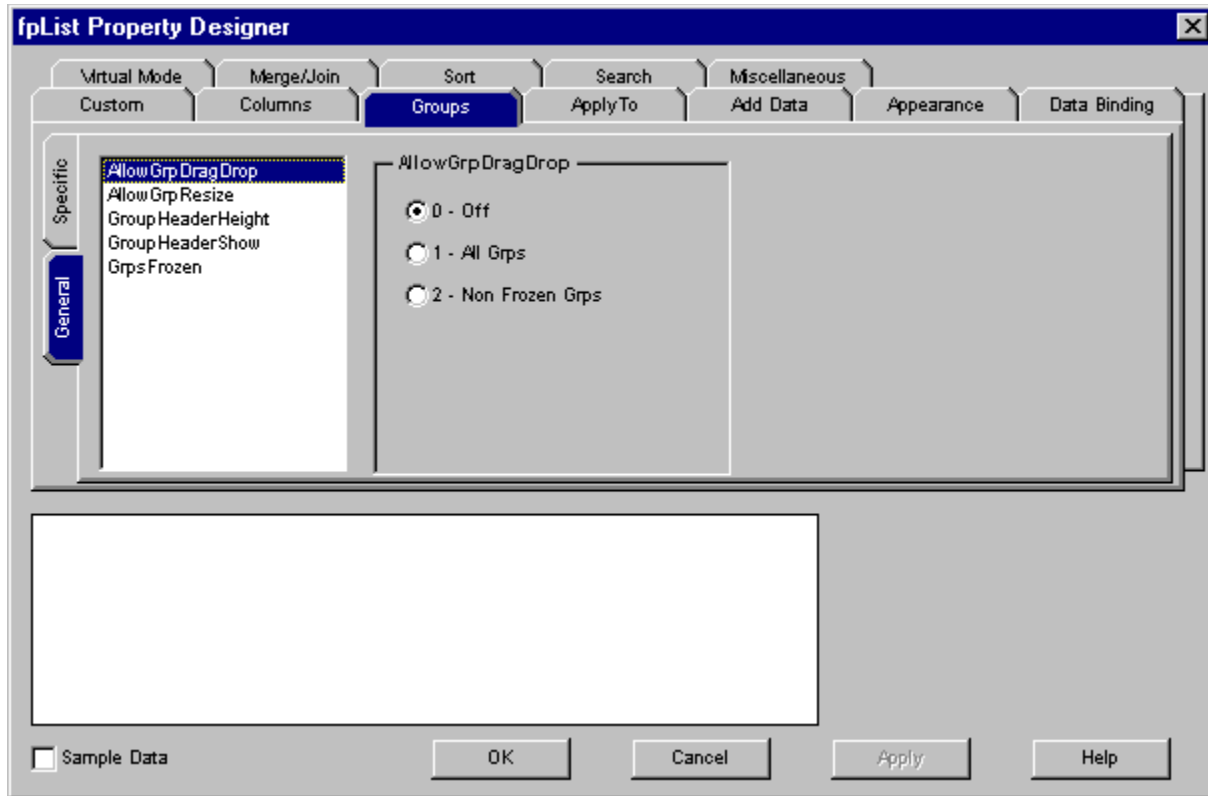
Groups
 [0]

[] Sample Data

OK | Cancel | Apply | Help

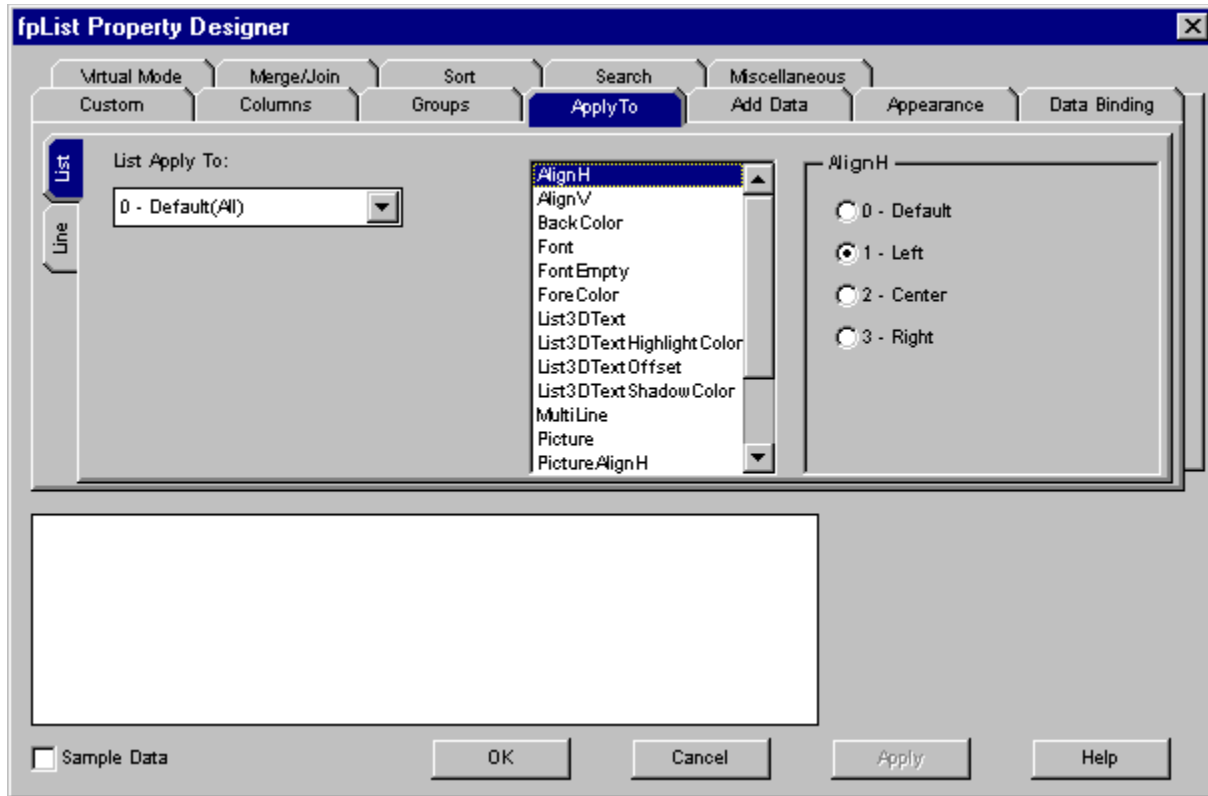
PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Groups **General Subtab**



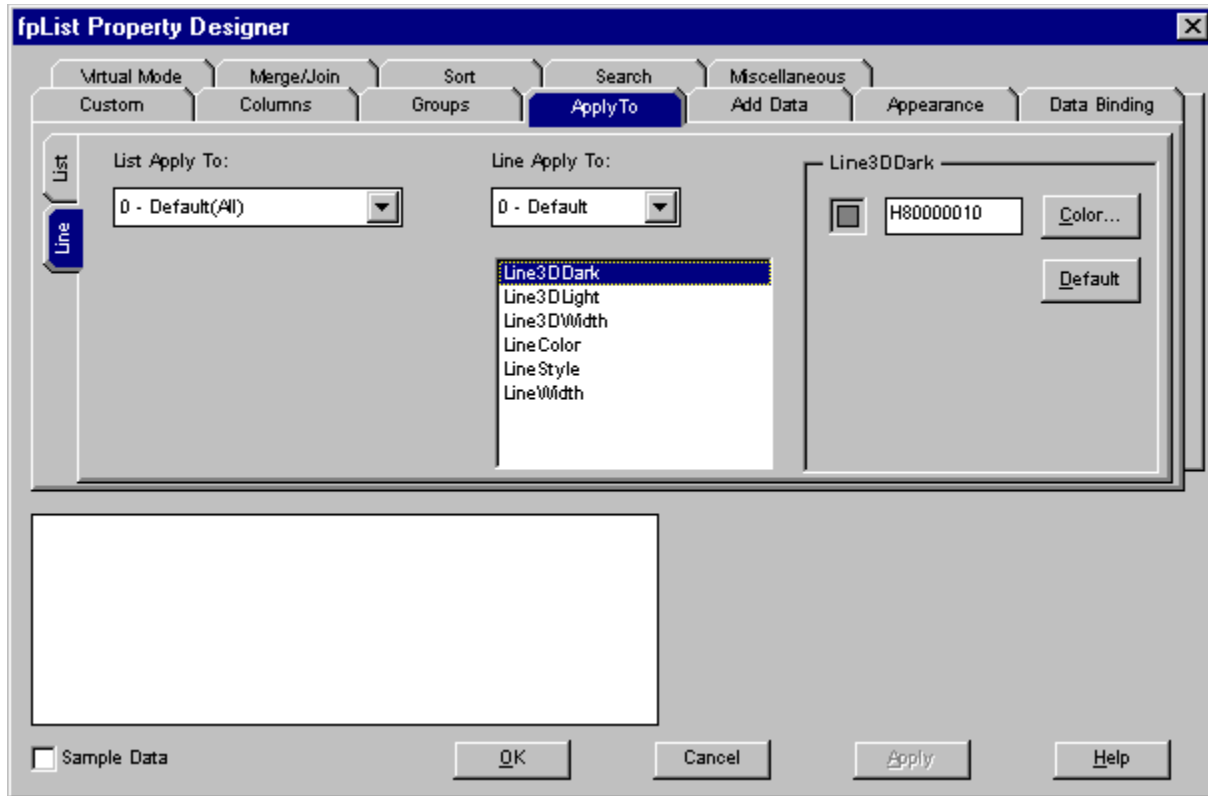
PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ApplyTo List Subtab



PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

ApplyTo Line Subtab



Add Data

The screenshot shows the 'fpList Property Designer' dialog box with the 'Add Data' tab selected. The dialog has a title bar with a close button (X) and a series of tabs: Virtual Mode, Merge/Join, Sort, Search, Miscellaneous, Custom, Columns, Groups, Apply To, Add Data (selected), Appearance, and Data Binding. The 'Add Data' tab contains the following elements:

- Individual** section with four input fields: Col, Col Name, Col ID, and Row. Each field has a dropdown arrow on its right side.
- Delete Row** button.
- A central list box containing the items: ColList (highlighted), InsertRow, List, and ListCount.
- A **Col List** section with a text input field.
- A large empty rectangular area at the bottom of the dialog.
- At the bottom of the dialog, there is a checkbox for **Sample Data** and four buttons: **OK**, **Cancel**, **Apply**, and **Help**.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Add Data fpCombo Control

fpCombo Property Designer [X]

Virtual Mode Merge/Join Sort Search Miscellaneous
 Custom Columns Groups Apply To **Add Data** Appearance Data Binding

Individual
 Col: []
 Col Name: []
 Col ID: []
 Row: []

[Delete Row]

ColList
 ColList
 InsertRow
 List
 ListCount
 Text

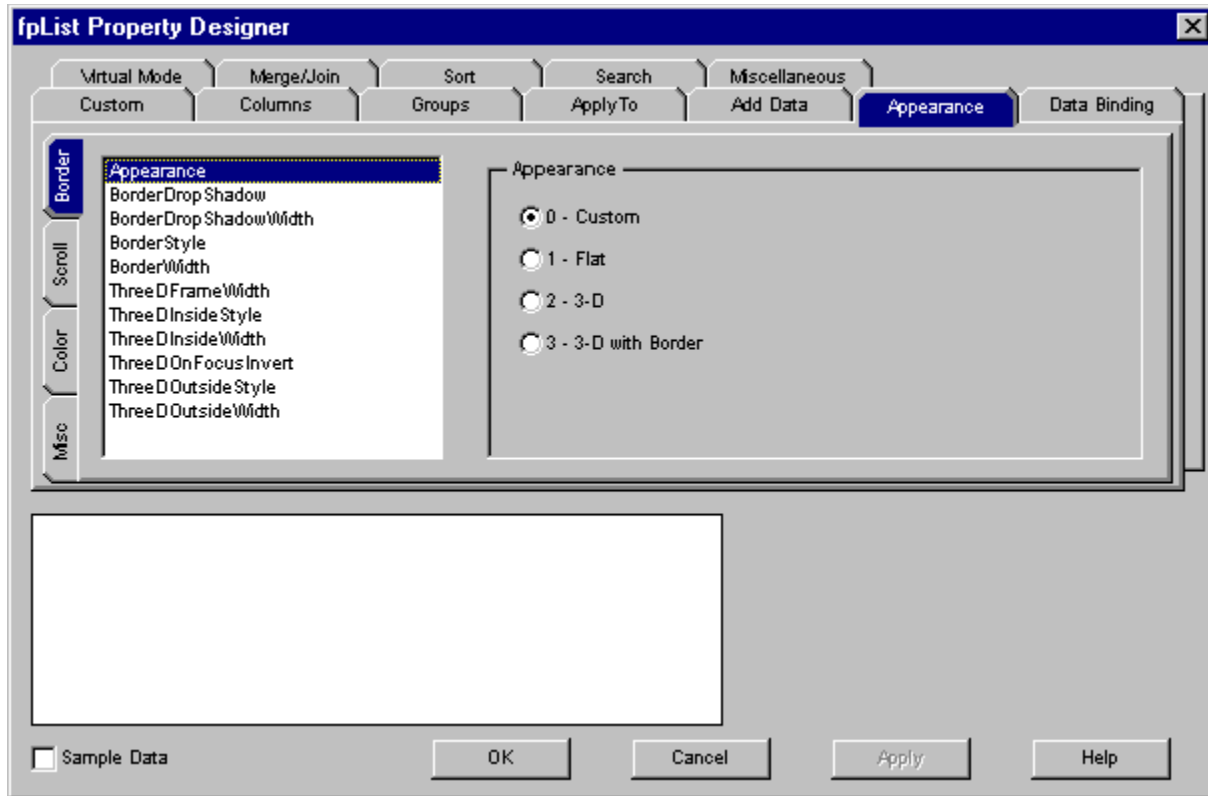
ColList
 []

[] []

Sample Data [OK] [Cancel] [Apply] [Help]

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Appearance **Border Subtab**

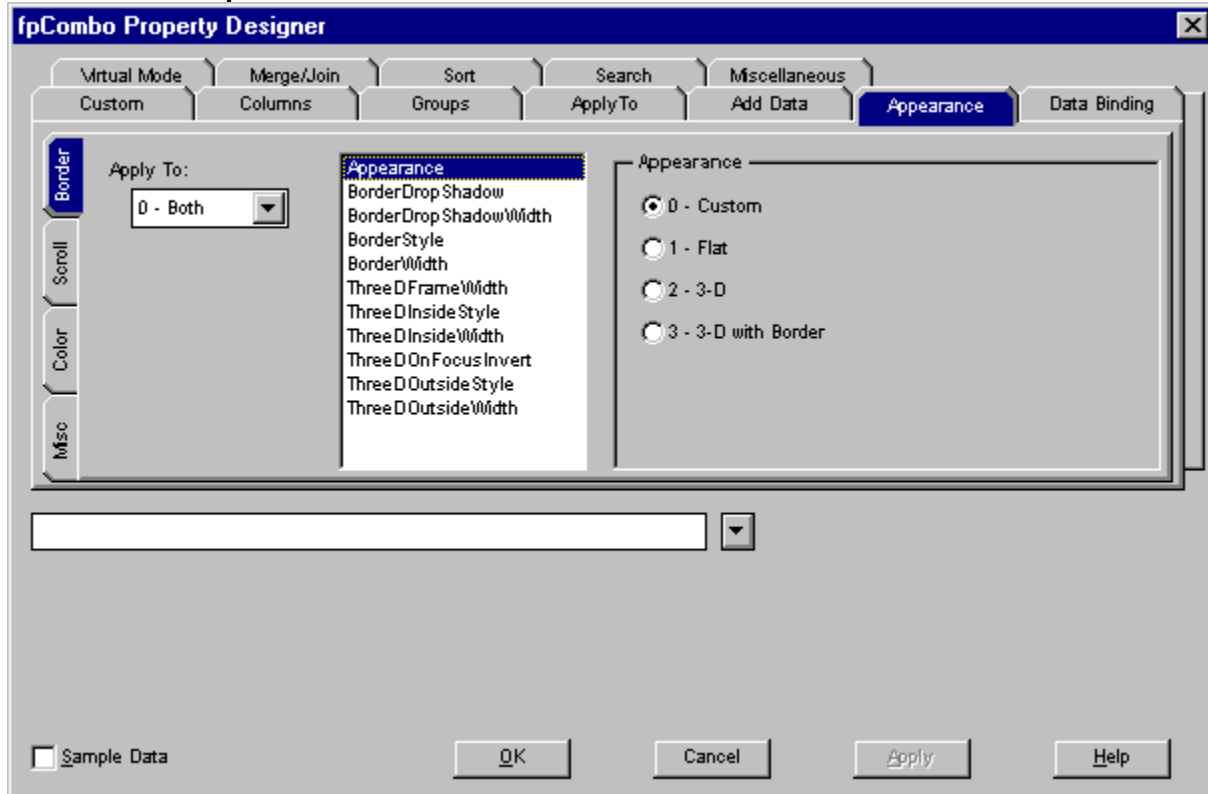


PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Appearance Border Subtab

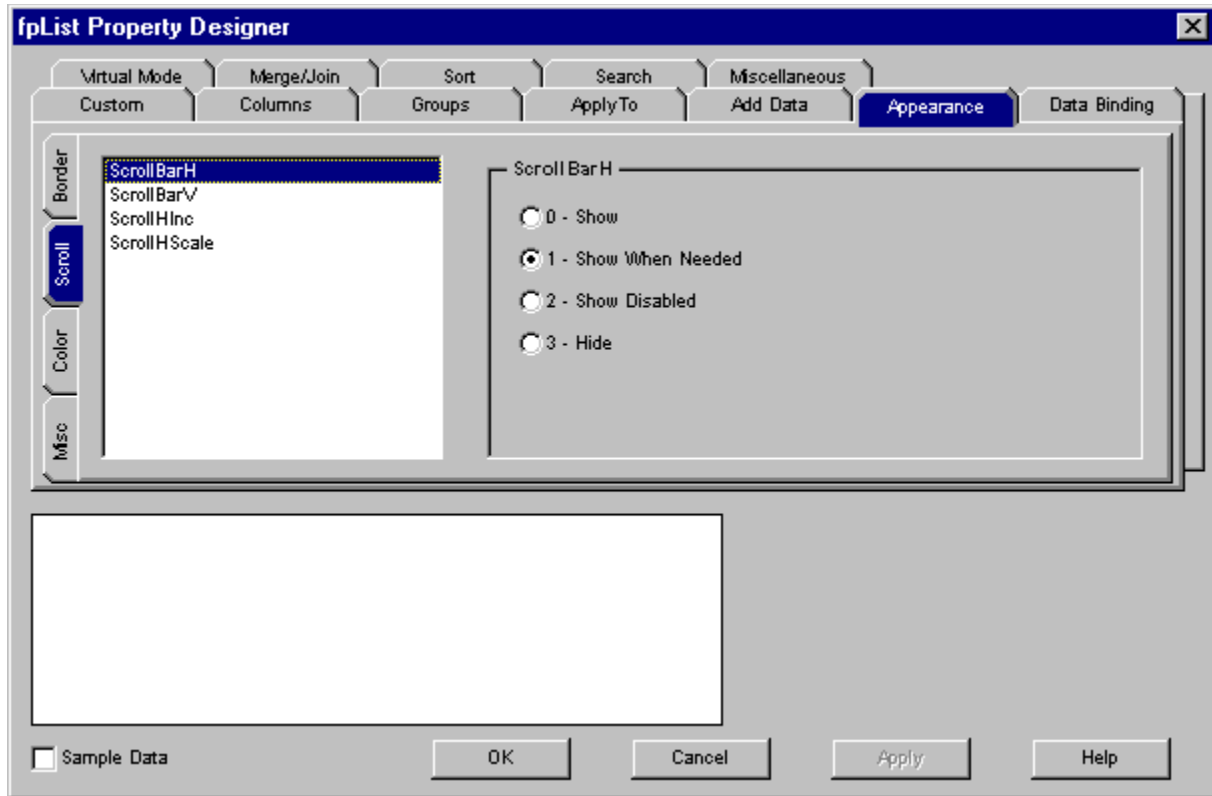
PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

fpCombo Control



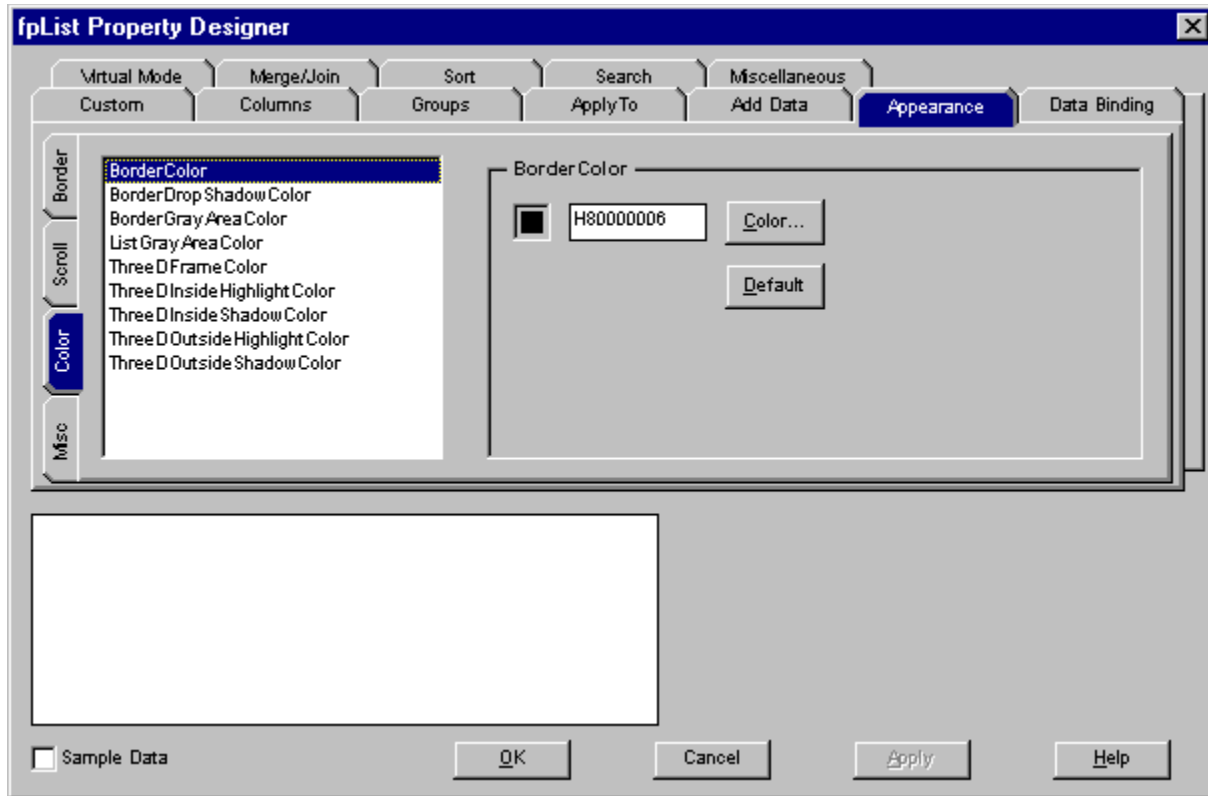
PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Appearance Scroll Subtab



PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Appearance Color Subtab

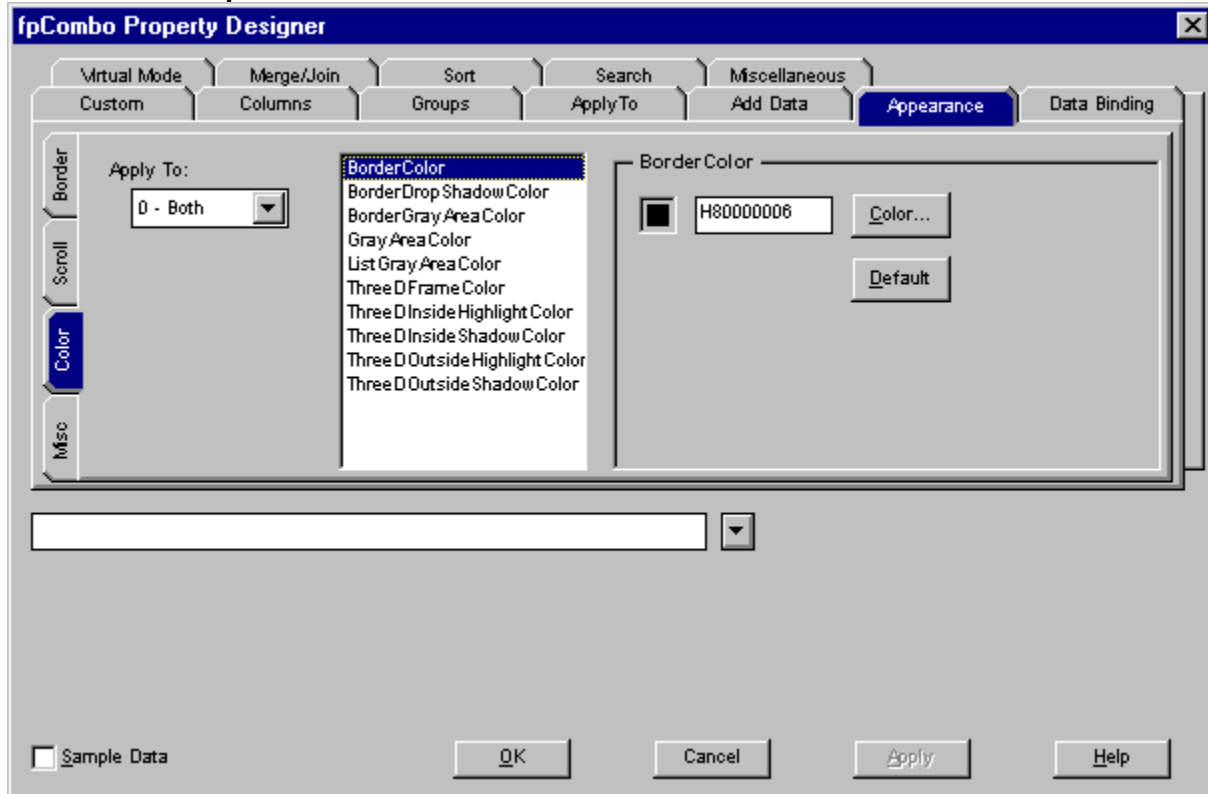


PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Appearance Color Subtab

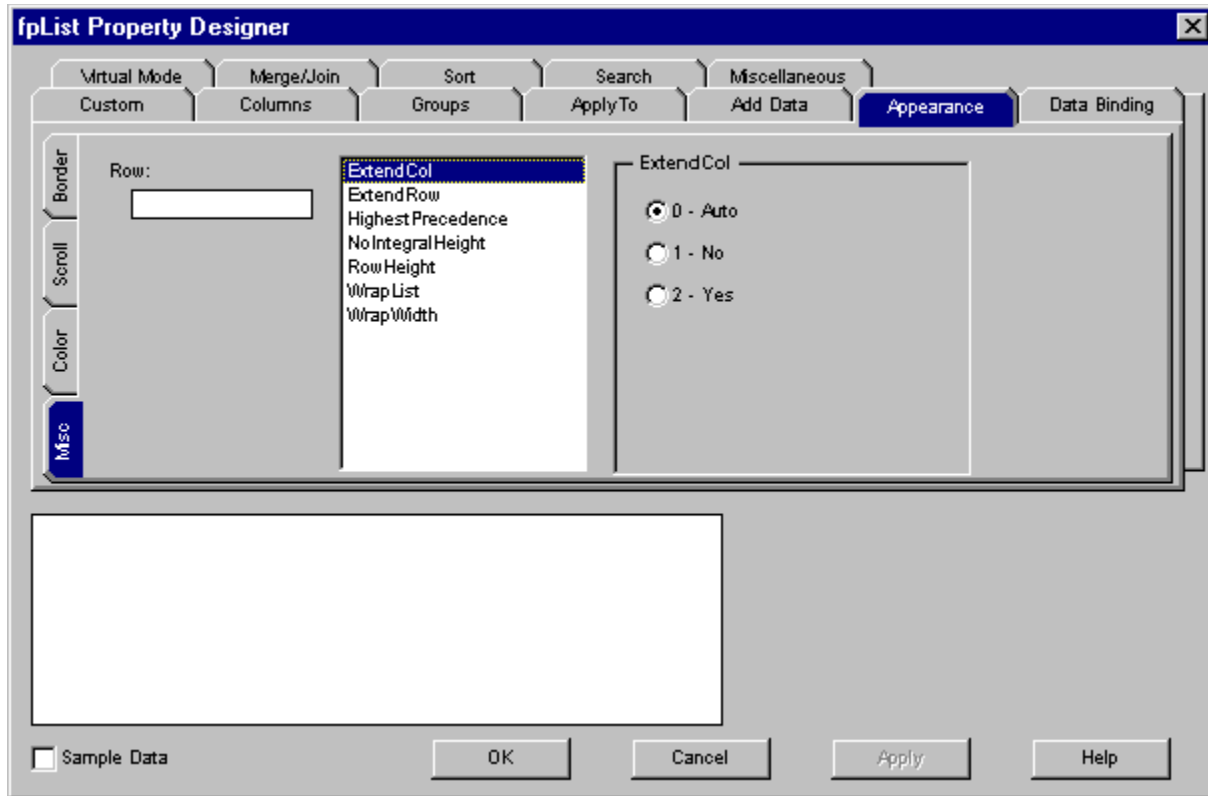
PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

fpCombo Control



PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Appearance Misc Subtab

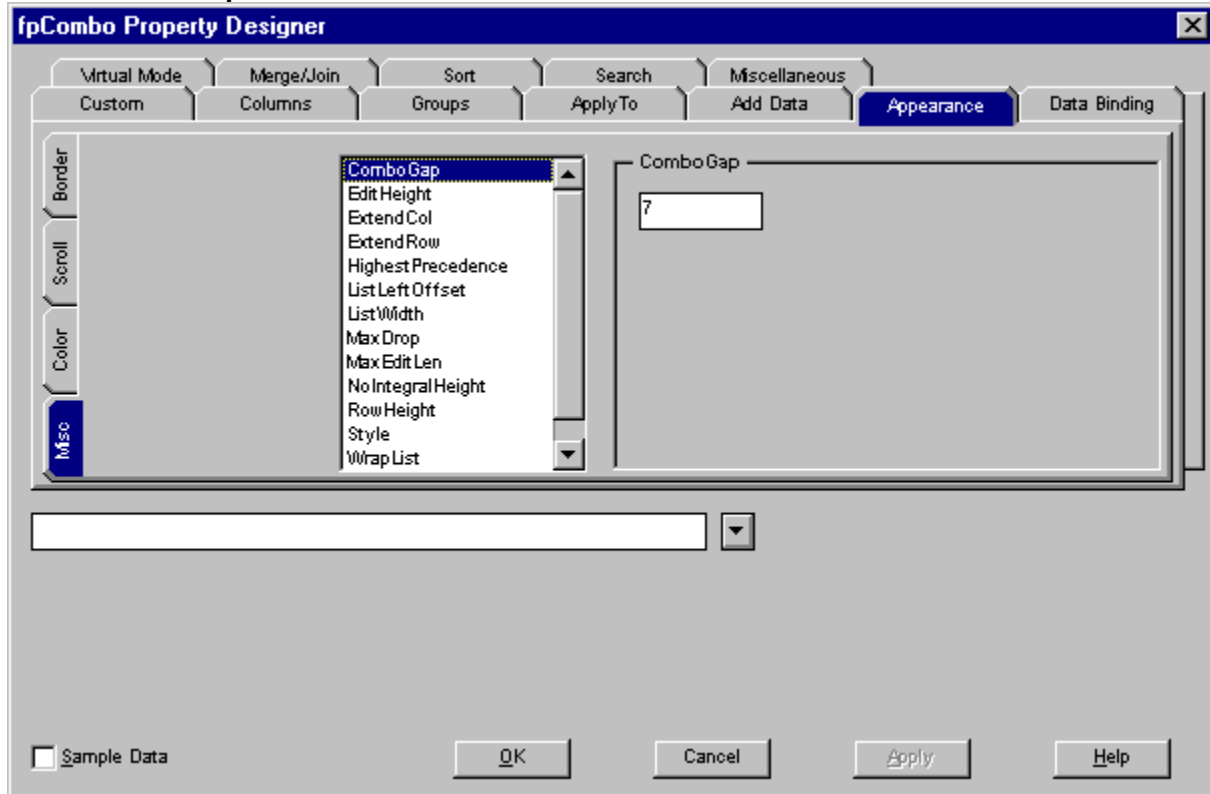


PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Appearance Misc Subtab

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

fpCombo Control



PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Data Binding **Column Subtab**

fpList Property Designer [X]

Virtual Mode | Merge/Join | Sort | Search | Miscellaneous
 Custom | Columns | Groups | Apply To | Add Data | Appearance | **Data Binding**

Columns
 Individual
 Col: []
 Col Name: []
 Col ID: []

General

Col DataField
 ColFormat

Col DataField []
 []

Sample Data

[OK] [Cancel] [Apply] [Help]

	PD	RD	WR	RT	DT
✓	✓	✓	✓	✓	✓

Data Binding Column Subtab

	PD	RD	WR	RT	DT
✓	✓	✓	✓	✓	✓

fpCombo Control

fpCombo Property Designer [X]

Virtual Mode Merge/Join Sort Search Miscellaneous
 Custom Columns Groups Apply To Add Data Appearance **Data Binding**

Columns
 Individual
 Col: []
 Col Name: []
 Col ID: []

General

Col DataField
 ColFormat
 ColumnBound
 ColumnEdit

Col DataField []
 []

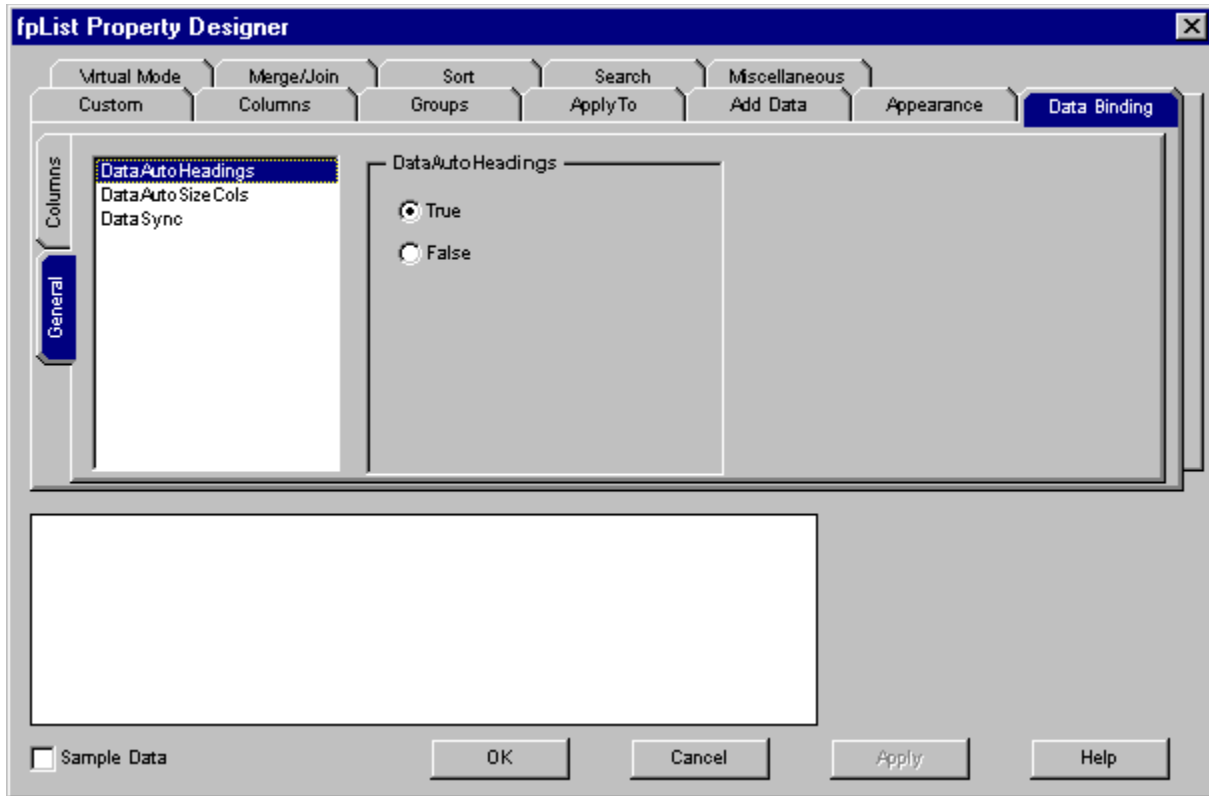
[] []

Sample Data

[OK] [Cancel] [Apply] [Help]

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Data Binding **General Subtab**



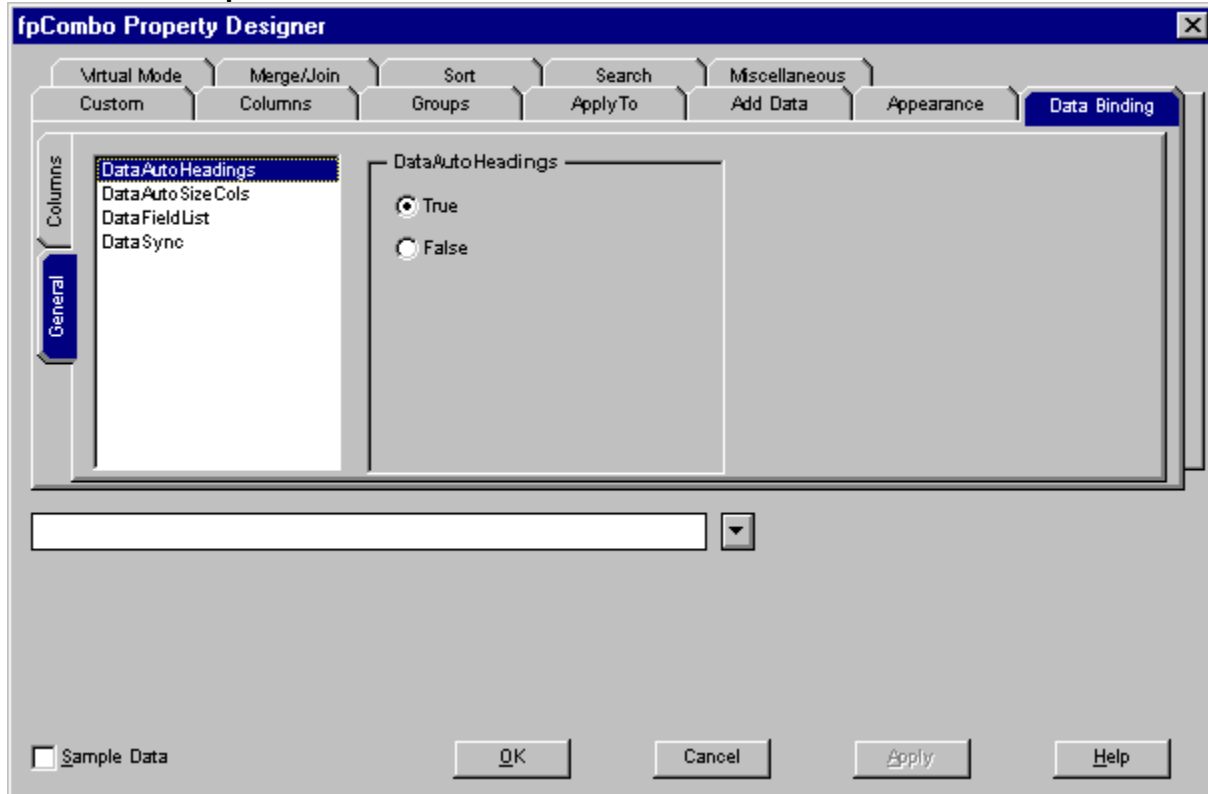
Data Binding

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

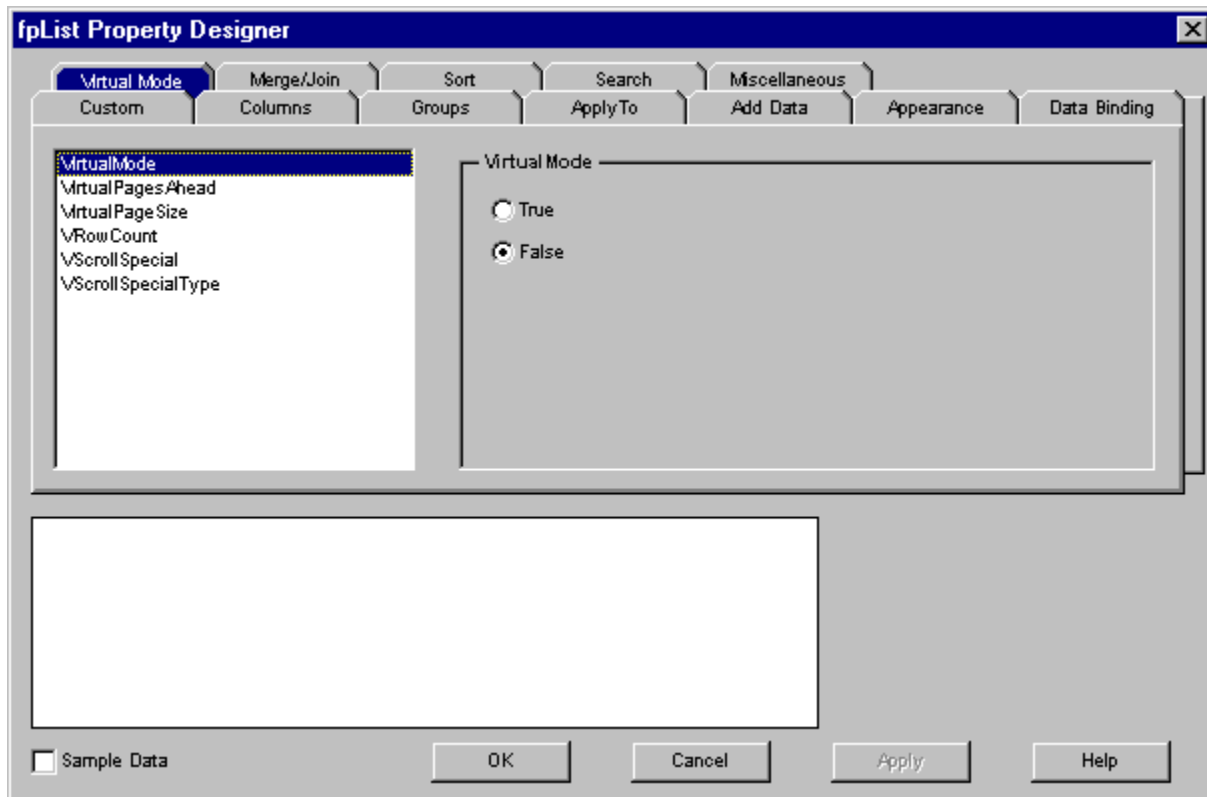
 General Subtab

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

 fpCombo Control



Virtual Mode



Merge/Join

The screenshot shows the 'fpList Property Designer' dialog box with the 'Merge/Join' tab selected. The dialog has a title bar with a close button (X) and a menu bar with options: Virtual Mode, Merge/Join, Sort, Search, Miscellaneous, Custom, Columns, Groups, Apply To, Add Data, Appearance, and Data Binding.

The 'Merge/Join' tab contains the following elements:

- Individual:** A group box containing four input fields: 'Col:', 'Col Name:', 'Col ID:', and 'Row:'. Each field has a dropdown arrow on its right side.
- List:** A list box containing the following items: 'ColMerge' (highlighted), 'JoinID', 'MergeAdjustView', and 'RowMerge'.
- Col Merge:** A group box containing three radio button options:
 - 0 - Off
 - 1 - Always
 - 2 - Restricted

At the bottom of the dialog, there is a large empty rectangular area, a checkbox labeled 'Sample Data', and four buttons: 'OK', 'Cancel', 'Apply', and 'Help'.

Sort

The image shows the 'fpList Property Designer' dialog box with the 'Sort' tab selected. The dialog has a title bar with a close button. Below the title bar are several tabs: 'Virtual Mode', 'Merge/Join', 'Sort', 'Search', and 'Miscellaneous'. Underneath these are sub-tabs: 'Custom', 'Columns', 'Groups', 'Apply To', 'Add Data', 'Appearance', and 'Data Binding'. The 'Sort' tab is active, showing an 'Individual' section with three dropdown menus for 'Col:', 'Col Name:', and 'Col ID:'. A list box contains 'ColSort DataType', 'ColSortSeq', 'Sorted', and 'SortState', with 'ColSort DataType' selected. To the right, the 'Col Sort DataType' section has four radio buttons: '0 - Text (No Case)' (selected), '1 - Text (Case)', '2 - Integer', and '3 - Float'. At the bottom left is a 'Sample Data' checkbox. At the bottom right are 'OK', 'Cancel', 'Apply', and 'Help' buttons.

fpList Property Designer

Virtual Mode Merge/Join **Sort** Search Miscellaneous

Custom Columns Groups Apply To Add Data Appearance Data Binding

Individual

Col: []

Col Name: []

Col ID: []

ColSort DataType
ColSortSeq
Sorted
SortState

Col Sort DataType

0 - Text (No Case)

1 - Text (Case)

2 - Integer

3 - Float

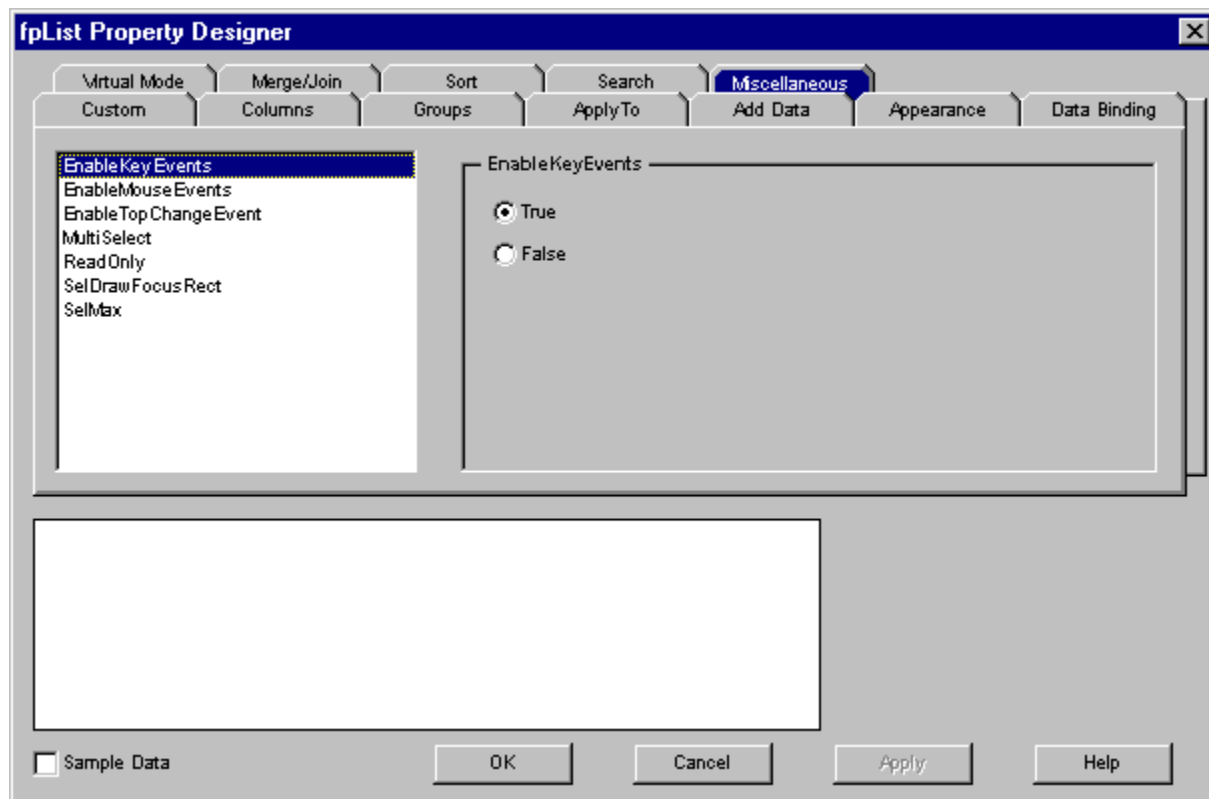
Sample Data

OK Cancel Apply Help

Search

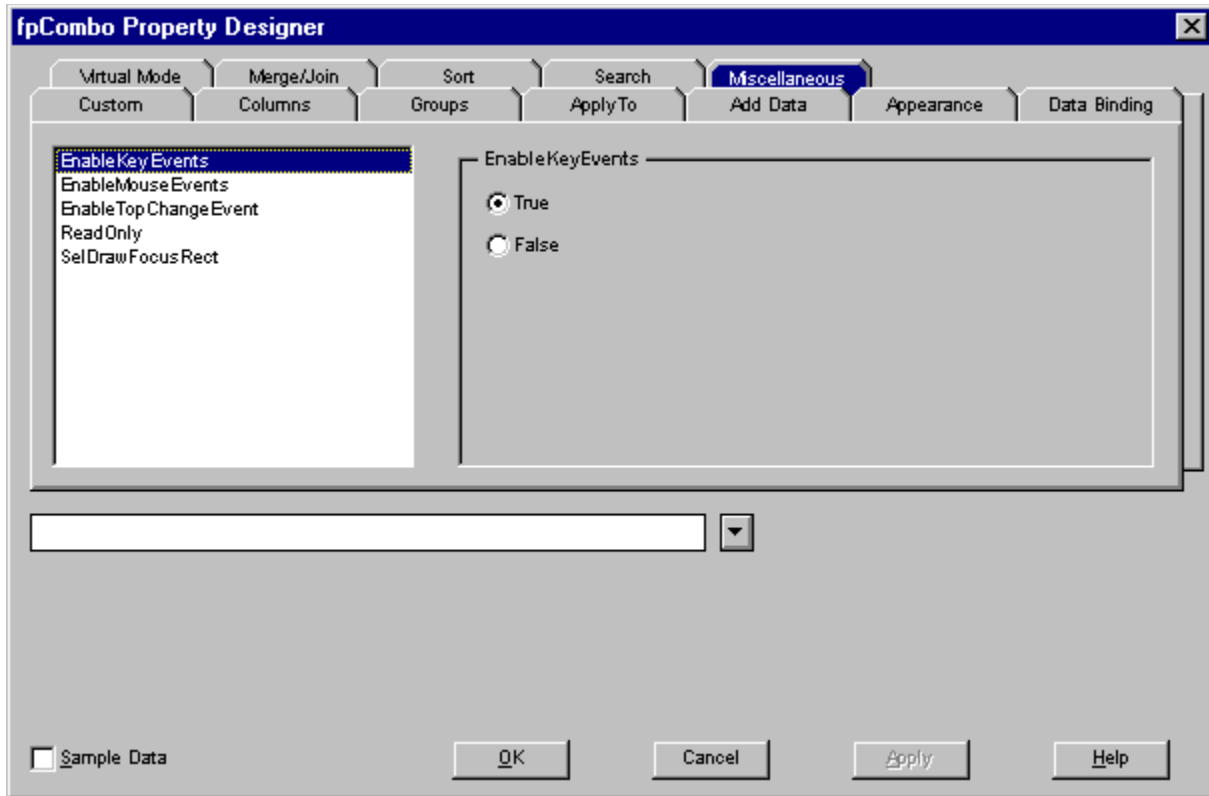
PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Miscellaneous



PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Miscellaneous fpCombo Control



Working with Columns

For an overview about columns and how they work, see [Columns, Rows, and Cells](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating Multiple Columns](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Referencing a Column](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Applying Properties to a Specific Column](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating Levels of Columns Within a Row](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Making a Column a Child of a Group](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Providing Column Headers](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Resizing Columns](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Customizing Columns](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Moving Columns in the Control](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Cloning Columns](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Joining Cells](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Merging Columns or Rows](#)

Creating Multiple Columns

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

A multiple column fpCombo or fpList control has a list that contains one or more columns.

The edit field of a multiple-column fpCombo control displays all the values in the selected row, separated by the column separator character (the default is a tab character). For best readability, display one column in the edit field.

If you have multiple columns, you can display those columns on different levels within a row. A column can also occupy more than one level in a row. For more information on creating multiple levels in a row, see [Creating Levels of Columns Within a Row](#).

[Print](#)

[Copy](#)

[Close](#)

To create a multiple-column fpCombo or fpList control

Designer Page

1. Specify the number of columns in the control.
 - a. On the [Specific subtab of the Columns designer page](#), select the [Columns](#) property from the properties list box.
 - b. Type the number of columns in the box under Columns in the property value area.
2. If you want to display only one column in the edit field,
 - a. On the [Custom designer page](#), select the [ColumnEdit](#) property from the properties list box.
 - b. Type the column number in the box under ColumnEdit in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To create a multiple-column fpCombo or fpList control

Browser/Code

1. Set the [Columns](#) property to the number of columns you want to create.
2. If you want to display only one column in the edit field, specify the column with the [ColumnEdit](#) property.

Referencing a Column

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

List Pro controls have three methods to reference a specific column: index number, name, and identifier number.

Column index numbers are assigned by the control and are based on the physical position of the column in the control. These numbers are zero-based, and columns are numbered from left to right and top to bottom within their parent group, if any, and then within the control. If you move or insert a column, the control changes the index numbers. Use the [Col](#) property to reference columns by their index numbers.

You can also define a unique name or identifier number for each column. If you use a column name or identifier number to specify a column, the properties you set for a particular column are applied to that column, regardless of its position in the control. Use the [ColName](#) property to define column names or the [ColID](#) property to define column identifier numbers. Typically, you should use either the name or the identifier number, but not both.

For example, assume you define four columns in your control as follows.

Col	ColID	ColName
0	1	Monday
1	2	Tuesday
2	3	Wednesday
3	4	Thursday

The control appears as shown below.

```
bmc 5colref1.mrb}
```

As the first example, assume you move the third column (Col = 2) to the far left side of the control. The control now appears as shown below.

Col = 0, ColID = 3, ColName = Wednesday	Col = 1, ColID = 1, ColName = Monday	Col = 2, ColID = 2, ColName = Tuesday	Col = 3, ColID = 4, ColName = Thursday
--	---	--	---

Note that the column index numbers changed but the column identifier numbers and column names did not change.

As the second example, assume you create two parent groups and you make the first, second, and fourth columns children of the first group. You make the third column a child of the second group. Also assume the fourth column is put on the second level. The control appears as shown below.

Group 0		Group 1
Col = 0, ColID = 1, ColName = Monday	Col = 1, ColID = 2, ColName = Tuesday	Col = 3, ColID = 3, ColName = Wednesday
Col = 2, ColID = 4, ColName = Thursday		

Note that the column index numbers changed and are numbered from left to right and top to bottom within their parent group.

Once you have specified the number of columns in the control and created identifier numbers or names, you can specify custom features for each column individually. For more information on setting properties for specified columns, see [Applying Properties to a Specific Column](#).

[Print](#)

[Copy](#)

[Close](#)

To reference a column

Designer Page

1. Specify the column. On the [Specific subtab of the Columns designer page](#), under Individual, select the column number from the Col drop-down list box.
2. If you want to define a column identifier number,
 - a. Select the [ColID](#) property from the properties list box.
 - b. Type the identification number in the box under ColID in the property value area.
3. If you want to define a column name,
 - a. Select the [ColName](#) property from the properties list box.
 - b. Type the name in the box under ColName in the property value area.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

4. Repeat steps 1-3 until all columns have been defined.
5. To specify the column to which an action or property applies, either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.

[Print](#)

[Copy](#)

[Close](#)

To reference a column

Code

1. Specify the column with the [Col](#) property.
2. If you want to define a column identifier number, set the [ColID](#) property.
3. If you want to define a column name, set the [ColName](#) property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

4. Repeat steps 1

✓	✓	✓	✓	✓
---	---	---	---	---

 3 until all columns have been defined.
5. Specify the column to which an action or property applies with the Col, [ColFromID](#), or [ColFromName](#) property.

Applying Properties to a Specific Column

Once you have specified the number of columns in the control and defined the column identifier number or name, you can use the designated-column properties to perform an action on a specific column. Designated-column properties require that the [Col](#), [ColFromID](#), or [ColFromName](#) property be set before using them.

The following table contains designated-column properties for the List Pro controls.

ColDataField	ColList	ColSortDataType
ColFormat	ColLockResize	ColSorted
ColHeaderText	ColMerge	ColSortSeq
ColHide	ColParentGroup	ColText
ColLevel	ColPos	ColWidth
ColLevelHeight	ColPosInParent	

If you want to use the designated-column properties in a List Pro control with only one column, you must set the [Columns](#) property to 1 and the Col property to 0.

Creating Levels of Columns Within a Row

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

Rows in a List Pro control can have multiple levels of columns. The number of levels you set applies to every row in the control. For example, if you want two levels in one row in an fpList control, every row in the control has to have two levels.

Row 0	Level 0	Dixie	75
	Level 1	Weimaraner	
		Tasha	65
		Labrador	
		Kamala	90
		Labrador	

For each column in a row, you can specify the level number and the height in levels. The height cannot exceed the total number of levels in a row. The height of the column in levels applies to the same column in every row. For example, if you define column 1 to be two levels high, column 1 is two levels high in every row. In the preceding example, each column is one level high.

The height of each level is equal to the row height. For example, if the control has three levels and the row height is set to 100 twips. Each level within the row is 100 twips high.

If you display column headers and you have multiple levels in the control, the column headers also appear on multiple levels.

column header (2 levels)		Breed	Weight	column header (1 level)
		Labrador	90	
column (2 levels)		Rottweiler	125	column (1 level)
		Weimaraner	75	

To summarize, creating levels of columns is a three step process:

1. Set the number of levels in rows.
2. Set up which columns are on which levels.
3. Specify the height of each column in levels.

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Create and define all columns and column references (using the [Col](#), [ColID](#), and [ColName](#) properties) in the control before moving columns with the [ColLevel](#) property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Because column index numbers are based on the physical position of the column in the control, if you plan on changing column levels, you should use one of the unique column identifiers (ColID or ColName property) to reference a column rather than the Col property.

[Print](#)

[Copy](#)

[Close](#)

To create levels of columns within a row

Designer Page

1. Specify the number of levels in every row of the control.
 - a. On the [General subtab of the Columns designer page](#), select the [ColumnLevels](#) property from the property list box.
 - b. Type the number of levels in the box under ColumnLevels in the property value area.
2. Specify the column. On the [Specific subtab of the Columns designer page](#), either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.
3. Specify the level of the column.
 - a. Select the [ColLevel](#) property from the properties list box.
 - b. Type the level number in the box under ColLevel in the property value area.
4. Specify the level height for the column.
 - a. Select the [ColLevelHeight](#) property from the properties list box.
 - b. Type the level height in the box under ColLevelHeight in the property value area.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

5. Repeat steps 2

✓	✓	✓	✓	✓
---	---	---	---	---

 4 until all column levels are defined.

[Print](#)

[Copy](#)

[Close](#)

To create levels of columns within a row

Browser/Code

1. Specify the number of levels in every row of the control with the [ColumnLevels](#) property.
2. At run time, specify the column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.
3. At run time, specify the level of the column with the [ColLevel](#) property.
4. At run time, specify the height of the column with the [ColLevelHeight](#) property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

5. Repeat steps 2-4 until all column levels are defined.

Making a Column a Child of a Group

You can make a column a child of a group. For example, you might want to make all the address columns, such as street information, city, state, and zip code, children of an address group.

Children of groups exhibit the following characteristics:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If you move a group, the groups children move with it. For more specific information, see [Moving](#)

[Groups in the Control](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

When you hide a group, the groups children are also hidden.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Children are automatically sized to fit the group width. This can result in text not being fully displayed in a column. For more information, see [Calculating the Width of Group Children](#).

Groups can have other groups as children or columns as children, but not a combination of the two.

For information on how to make a column a child of a group, see [Creating Children of Groups](#).

Providing Column Headers

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating Column Headers](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Customizing Column Headers](#)

Creating Column Headers

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can provide and customize column headers for the list in List Pro controls. A header contains static text that does not scroll vertically with the list. Headers can be useful for labeling multiple-column lists.

Group headers can also serve as column headers. You can display both column and group headers.

Note By default, when each column of a multiple-column fpCombo or fpList control is bound to a database, the header displays the name of the associated database field. You can display other header text by following this procedure.

[Print](#)

[Copy](#)

[Close](#)

To create column headers

Designer Page

1. If you are working with a single-column fpCombo or fpList control,
 - a. On the [Specific subtab of the Columns designer page](#), select the [Columns](#) property in the properties list box.
 - b. Type 1 in the box under Columns in the property value area.
2. Specify the column. Either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.
3. Specify the header text for each column.
 - a. Select the [ColHeaderText](#) property from the properties list box.
 - b. Type the header text in the box under ColHeaderText in the property value area.
4. Repeat steps 2 and 3 until you have specified the header text for all the column headers.
5. Display the column headers.
 - a. On the [General subtab of the Columns designer page](#), select the [ColumnHeaderShow](#) property in the properties list.
 - b. Select the True option button under ColumnHeaderShow in the property value area.
6. Specify the column header height.
 - a. Select the [ColumnHeaderHeight](#) property in the properties list.
 - b. Type the header height in twips in the box under ColumnHeaderHeight.
If the header text is long, you can wrap the text to multiple lines. For more information, see [Wrapping Text in a List Pro Control](#).
7. If you are working with a bound fpCombo or fpList control, turn off automatic headers.
 - a. On the [General subtab of the Data Binding designer page](#), select the [DataAutoHeadings](#) property in the properties list.
 - b. Select the False option button under DataAutoHeadings in the property value area.

Note If you are working with a bound control with one column, do not set the DataField property for the single column. Make sure the Columns property is set to 1 and specify the field to bind to the column using the [ColDataField](#) property.

[Print](#)

[Copy](#)

[Close](#)

To create column headers

Browser/Code

1. If you are working with a single-column fpCombo or fpList control, set the [Columns](#) property to 1.
2. Set the [ColumnHeaderShow](#) property to True.
3. Specify the column header height in twips with the [ColumnHeaderHeight](#) property.
If the header text is long, you can wrap the text to multiple lines. For more information, see [Wrapping Text in a List Pro Control](#).
4. If you are working with a bound fpCombo or fpList control, set the [DataAutoHeadings](#) property to False.

Note If you are working with a bound control with one column, do not set the DataField property for the single column. Make sure the Columns property is set to 1 and specify the field to bind to the column using the [ColDataField](#) property.

5. At run time, specify the header text for each column by performing the following actions:
 - a. Specify the column for which to set the header text with the [Col](#), [ColFromID](#), or [ColFromName](#) property.
For single-column fpCombo and fpList controls, set the Col property to 0.
 - b. Specify the header text with the [ColHeaderText](#) property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

- c. Repeat steps 5.a 5.b until you have specified the header text for all the columns.

Customizing Column Headers

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

You can customize the way column headers are displayed in an fpCombo or fpList control. You can customize characteristics such as displaying and aligning pictures, changing text color and appearance, displaying lines, and changing the background color.

[Print](#)

[Copy](#)

[Close](#)

To customize column headers

Designer Page

1. Specify which header or headers to customize.
 - a. On the [List subtab of the ApplyTo designer page](#), choose either 7 - Col Headers or 9 - Single Col Header from the List Apply To drop-down list box.
 - b. If you are customizing an individual column header, specify the column. Either select the column from either the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.
2. Follow the instructions in the appropriate topic listed in the following table.

For more information on . . .

Pictures and picture alignment

Text color, appearance, and alignment

Lines

Background color

See . . .

[Working with Text and Graphics](#)

[Working with Text and Graphics](#)

[Customizing Lines](#)

[Changing the Background Color](#)

[Print](#)

[Copy](#)

[Close](#)

To customize column headers

Code

1. Set the [ListApplyTo](#) property to 7 (Col Headers) or 9 (Single Col Header).
2. If you are customizing an individual column header, specify the column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.
3. Follow the instructions in the appropriate topic listed in the following table.

For more information on . . .

Pictures and picture alignment

Text color, appearance, and alignment

Lines

Background color

See . . .

[Working with Text and Graphics](#)

[Working with Text and Graphics](#)

[Customizing Lines](#)

[Changing the Background Color](#)

Resizing Columns

You can use either of the following methods to change the width of a column:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Use the mouse](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Specify a column width](#)

Using the Mouse to Resize Columns


PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

By default, you cannot resize columns using the mouse. With the [AllowColResize](#) property, you can resize any unlocked column by dragging the column or header boundary to a new position with the resize pointer . For more information about locking specific columns against resizing, see [Customizing Columns](#).

[Print](#)

[Copy](#)

[Close](#)

To resize columns using the mouse

Designer Page

1. Specify how columns are resized.
 - a. On the [General subtab of the Columns designer page](#), select the [AllowColResize](#) property from the properties list box.
 - b. Select either the 1 - Resize Header or 2 - Resize Col or Header option button in the property value area.
2. In the preview area,
 - a. Position the mouse pointer over the column boundary (if you chose option button 2 -Resize Col or Header) or header boundary of the column you want to move.
 - b. When the mouse pointer changes to the resize pointer, press the left mouse button and drag the boundary to resize the column.

[Print](#)

[Copy](#)

[Close](#)

To resize columns using the mouse

Browser/Code

1. Set the [AllowColResize](#) property to either 1 (Resize Header) or 2 (Resize Col or Header).
2. At run time,
 - a. Position the mouse pointer over the column boundary (if you set the AllowColResize property to 2 (Resize Col or Header)) or header boundary of the column you want to move.
 - b. When the mouse pointer changes to the resize pointer, press the left mouse button and drag the boundary to resize the column.

Specifying the Column Width

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can set the width of a column in a List Pro control to a specific value. The default unit of measurement for column width is the average character width of the current font. However, you can specify the width in twips, pixels, or the maximum character width of the current font.

Note When columns are grouped, the column width is determined by the width of the group and other group children (see [Calculating the Width of Group Children](#)). If you want to set the width of a column that is a child of a group, use the following procedure after making the column a child of the group.

[Print](#)

[Copy](#)

[Close](#)

To specify the column width

Designer Page

1. If you want to specify a different unit of measurement,
 - a. On the [General subtab of the Columns designer page](#), select the [ColumnWidthScale](#) property from the properties list box.
 - b. Select the appropriate option button under ColumnWidthScale in the property value area.
2. Specify the column. On the [Specific subtab of the Columns designer page](#), either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.
3. Specify the column width.
 - a. Select the [ColWidth](#) property from the properties list box.
 - b. Type the width in the box under ColWidth in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To specify the column width

Browser/Code

1. If you want to specify a different unit of measurement, set the [ColumnWidthScale](#) property.
2. At run time,
 - a. Specify the column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.
 - b. Set the [ColWidth](#) property.

Customizing Columns

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can customize the columns of a List Pro control. You can:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Hide a column in the list

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Freeze a specific number of leftmost columns from scrolling horizontally, creating row headers

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Prevent a specific column from being resized using the mouse

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Specify a format string for displaying bound data

You can also customize column attributes as listed in the following table.

For more information on . . .

Pictures and picture alignment
Text color, appearance, and alignment
Lines
Background color

See . . .

[Working with Text and Graphics](#)
[Working with Text and Graphics](#)
[Customizing Lines](#)
[Changing the Background Color](#)

[Print](#)

[Copy](#)

[Close](#)

To customize columns

Designer Page

1. If you want to hide a column in the list,
 - a. Specify the column. On the [Specific subtab of the Columns designer page](#), either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.
 - b. Hide the column.
 - i. Select the [ColHide](#) property from the properties list.
 - ii. Select the True option button under ColHide in the property value area.
2. If you want to prevent a set number of leftmost columns from scrolling horizontally,
 - a. On the [General subtab of the Columns designer page](#), select the [ColsFrozen](#) property from the properties list box.
 - b. Type the number of leftmost columns to remain frozen in the box under ColsFrozen in the property value area.
3. If you want to prevent a specific column from being resized using the mouse,
 - a. Specify the column. On the Specific subtab of the Columns designer page, either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.
 - b. Lock the column against resizing.
 - i. Select the [ColLockResize](#) property from the properties list box.
 - ii. Select the True option button from the property value area.
4. If you want to specify a format string for bound data,
 - a. Specify the column. On the Specific subtab of the Columns designer page or the Data Binding designer page, either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.
 - b. Define the format string.

- i. Select the [ColFormat](#) property from the properties list box.
- ii. Type the format string in the box under ColFormat in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To customize columns

Browser/Code

1. If you want to hide a column in the list, at run time,
 - a. Specify the column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.
 - b. Set the [ColHide](#) property to True.
2. If you want to prevent a set number of leftmost columns from scrolling horizontally, set the [ColsFrozen](#) property to the number of leftmost columns to remain frozen.
3. If you want to prevent a specific column from being resized using the mouse, at run time,
 - a. Specify the column you want to prevent from being resized with the [Col](#), [ColFromID](#), or [ColFromName](#) property.
 - b. Set the [ColLockResize](#) property to True.
4. If you want to specify a format string for bound data, at run time,
 - a. Specify the column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.
 - b. Set the [ColFormat](#) property.

Moving Columns in the Control

The position of a column is determined by its location in the control, including its location with respect to other columns, its location in a group, and its level within a row. You can use any of the following methods to move or position columns in a control:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Use the drag-and-drop method](#) (to change location with respect to other columns or in a group)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Specify the position of a column](#) using the [ColPos](#) or [ColPosInParent](#) property (to change location with respect to other columns or in a group)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Change the column level](#)

Using the Drag-and-Drop Method to Move Columns

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can allow movement of any nonfrozen column using the drag-and-drop method, assuming the control displays column headers. For more information on freezing columns, see [Customizing Columns](#). For more information on displaying column and group headers, see [Creating Column Headers](#).

When you move a column to the left of its current position, all columns to the right of the destination column up to the original position of the moved column will shift to the right. When you move a column to the right of its current position, all columns between the original position of the moved column and its destination position will shift to the left. When you move a column, the index number of any column affected by the move will change.

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

You must display column or group headers to move columns using the drag-and-drop method.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Because column index numbers are based on the physical position of the column in the control, if you allow columns to be moved, use one of the unique column identifiers ([ColID](#) or [ColName](#) property) to reference a column rather than the [Col](#) property.

For example, assume you have the following list box.

Col 1	Col 2	Col 3	Col 4
A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T
U	V	W	X

If you move Col 4 to the left of Col 2, both Col 2 and Col 3 are shifted to the right as shown in the following figure.

Col 1	Col 4	Col 2	Col 3
A	D	B	C
E	H	F	G
I	L	J	K
M	P	N	O
Q	T	R	S
U	X	V	W

Starting with the original layout, if you move Col 1 to the right of Col 3, both Col 2 and Col 3 are shifted to the left as shown in the following figure.

Col 2	Col 3	Col 1	Col 4
B	C	A	D
F	G	E	H
J	K	I	L
N	O	M	P
R	S	Q	T
V	W	U	X

If columns are in groups and you move a column from one group to another, the following occurs:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Columns in the destination group shift within the group to left or right as appropriate

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Columns in the original group are resized to fit the group width

Note You may want to resize the columns in the original and destination group after moving a column from one group to another group. For more information, see [Specifying the Column Width](#).

For example, assume you have the following list box.

Group 1		Group 2	
Col 1	Col 2	Col 3	Col 4
A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T

If you move Col 3 into Group 1 to the left of Col 1, Col 1 and Col 2 are shifted to the right in the destination group and Col 4 is resized to the width of Group 2.

Group 1			Group 2
Col 3	Col 1	Col 2	Col 4
C	A	B	D
G	E	F	H
K	I	J	L
O	M	N	P
S	Q	R	T

You can also move grouped columns by moving their parent group. For more information, see [Moving Groups in the Control](#).

[Print](#)

[Copy](#)

[Close](#)

To drag and drop columns

Designer Page

1. Specify which columns you can drag and drop.
 - a. On the [General subtab of the Columns property page](#), select the [AllowColDragDrop](#) property from the properties list box.
 - b. Select either the 1- All Cols or the 2 -Non Frozen Cols option button under AllowColDragDrop in the property value area.
2. In the preview area, position the mouse pointer over the header of the column you want to move.
3. When the mouse pointer changes to the drag-drop pointer (a hand), press the left mouse button and drag and drop the column in its new location.

[Print](#)

[Copy](#)

[Close](#)

To drag and drop columns

Browser/Code

1. Set the [AllowColDragDrop](#) property to either 1 (All Cols) or 2 (Non Frozen Cols).
2. At run time,
 - a. Position the mouse pointer over the header of the column you want to move.
 - b. When the mouse pointer changes to the drag-drop pointer (a hand), press the left mouse button and drag and drop the column in its new location.

Defining the Position of a Column Within the Control

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

You can move columns by defining their position within a List Pro control. If columns are grouped, you can also define the position of a column within a group.

Position numbers are zero-based and are numbered from left to right and top to bottom within a parent group, if any, and then within the control.

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Because column index numbers are based on the physical position of the column in the control, if you plan on moving columns, use one of the unique column identifiers ([ColID](#) or [ColName](#) property) to reference a column rather than the [Col](#) property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Define all columns (Col, ColID, and ColName properties) in the control before moving columns with the [ColLevel](#), [ColPos](#), or [ColPosInParent](#) property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If you have groups, you should use only the ColPosInParent property to position columns within groups. If you use the ColPos property and the ColPosInParent property at the same time, you might get unpredictable results.

[Print](#)

[Copy](#)

[Close](#)

To position a column in the control

Designer Page

1. Specify the column. On the [Specific subtab of the Columns designer page](#), either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.
2. If you want to specify the position of the column in the control,
 - a. Select the [ColPos](#) property in the properties list box.
 - b. Type the position number in the box under ColPos in the property value area.
3. If you want to specify the position of the column in its parent group,
 - a. Select the [ColPosInParent](#) property in the properties list box.
 - b. Type the position number in the box under ColPosInParent in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To position a column in the control

Code

1. Specify the column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.
2. If you want to specify the position of the column in the control, set the [ColPos](#) property.
3. If you want to specify the position of the column in its parent group, set the [ColPosInParent](#) property.

Cloning Columns

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

If you create a column that you would like to duplicate elsewhere in your control you can clone that column. All attributes of the original column except for data, column identification number ([ColID](#) property) or name ([ColName](#) property), and data field setting ([ColDataField](#) or DataField properties) are copied to the cloned column.

When you clone a column, the control inserts the cloned column to the right of the original column.

[Print](#)

[Copy](#)

[Close](#)

To clone a column

Designer Page

1. Specify the column you want to clone. On the [Specific subtab of the Columns designer page](#), either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.
2. Click the Clone Column button.

The cloned column is inserted to the right of the original column.

[Print](#)

[Copy](#)

[Close](#)

To clone a column

Code

At run time,

1. Specify the column you want to clone with either the [Col](#), [ColFromID](#), or [ColFromName](#) property.
2. Set the [Action](#) property to 12 (Clone Col).

The cloned column is inserted to the right of the original column.

Joining Cells

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

You can create a set of cells that are identical in content and appearance. The property characteristics of the first cell you assign to the joined set are applied to the other cells in the joined set. The advantages of joined cells are:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Joined cells look and function as one cell.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

A property set for one joined cell is applied to all joined cells.

To create joined cells, you assign the cells a joined identification number, therefore all cells within one control with the same joined identification number are in the same joined set. Joined identification numbers must be greater than zero.

Joined cells do not have to be adjacent to one another. However, if adjacent cells in a row are joined, the control merges the cells unless the rows have multiple levels. Adjacent cells in rows with multiple levels have the same content and appearance but the control does not merge them. For more information about merging cells, see [Merging Columns or Rows](#).

In the following figure, cells (0, 0), (0, 1), (1, 1), and (2, 2) are joined. Notice that cells (0, 1) and (1, 1) are adjacent and are merged.

	Col 0	Col 1	Col 2
A		B	C
A			F
G		H	A
J		K	L
M		N	O
P		Q	R

[Print](#)

[Copy](#)

[Close](#)

To join cells

Designer Page

1. Select the cell that you want to join.

On the [Merge/Join designer page](#), perform one of the following actions

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

number in the Row box.

Select the column from either the Col, Col Name, or Col ID drop-down list box and type the row

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Click the cell in the preview area.

2. Define the joined identification number.
 - a. Select the [JoinID](#) property in the properties list box.
 - b. Type the joined identification number in the box under JoinID in the property value area.
3. Repeat steps 1 and 2 until you have identified all joined cells.

[Print](#)

[Copy](#)

[Close](#)

To join cells

Code

1. Specify the cell you want to join with either the [Col](#), [ColFromID](#), or [ColFromName](#) property and the [Row](#) property.
2. Set the [JoinID](#) property.
3. Repeat steps 1 and 2 until you have identified all joined cells.

Merging Columns or Rows

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can merge cells in a column or row that contain the same text. You can specify that cells are always merged or that cells are merged only when adjacent cells to the left are also merged.

For example, an unmerged list box displays as shown in the following figure.

Day	Product	Line	Amount
Tuesday	Corn	C17	7.432
Tuesday	Rice	A23	2.302
Tuesday	Rice	C17	6.789
Monday	Rice	D14	5.432
Monday	Corn	C17	4.321
Monday	Beans	C17	4.321
Monday	Beans	A23	3.761

If you set the [RowMerge](#) property to 1 (Always) for columns 0, 1, and 2, the list box displays like this.

Day	Product	Line	Amount
Tuesday	Corn	C17	7.432
	Rice	A23	2.302
		C17	6.789
Monday	Rice	D14	5.432
		Corn	C17
	Beans	C17	4.321
		A23	3.761

Notice how the second product cell (Rice) merges across days to its left and lines to its right. You might not want these cells to merge that way.

If you set the [RowMerge](#) property to 2 (Restricted) for column 1, the list box displays like this.

Day	Product	Line	Amount
Tuesday	Corn	C17	7.432
	Rice	A23	2.302
		C17	6.789
Monday	Rice	D14	5.432
	Corn	C17	4.321
	Beans	C17	4.321
		A23	3.761

Notice how the Rice product cells no longer merge across days.

When you merge cells, cell characteristics (text appearance and background color, for example) from the upper-left cell in the merged block are applied to all merged cells within that block.

You can also specify that the cell contents of merged rows or columns are automatically centered in the portion of the cell that is displayed, vertically in the case of merged rows or horizontally in the case of merged columns. However, the cell contents do not scroll out of the viewing area until the entire row or column is out of the viewing area.

[Print](#)

[Copy](#)

[Close](#)

To merge columns or rows

Designer Page

1. If you want to merge cells in a column,
 - a. Specify the column. On the [Merge/Join designer page](#), either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.
 - b. Define how the column is merged.
 - i. Select the [ColMerge](#) property in the properties list box.
 - ii. Select either the 1 - Always or the 2 - Restricted option button under ColMerge in the property value area.
2. If you want to merge cells in a row,
 - a. Specify the row. On the Merge/Join designer page, either type the row number in the Row box or click the row.
 - b. Define how the row is merged.
 - i. Select the [RowMerge](#) property in the properties list box.
 - ii. Select either the 1 - Always or the 2 - Restricted option button under RowMerge in the property value area.
3. If you want to automatically center the text horizontally and vertically in the merged cells,
 - a. Select the [MergeAdjustView](#) property in the properties list box.
 - b. Select the True option button under MergeAdjustView in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To merge columns or rows

Browser/Code

1. If you want to merge cells in a column, at run time,
 - a. Specify the column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.
 - b. Set the [ColMerge](#) property to either 1 (Always) or 2 (Restricted).
2. If you want to merge cells in a row, at run time,
 - a. Specify the row with the [Row](#) property.
 - b. Set the [RowMerge](#) property to either 1 (Always) or 2 (Restricted).
3. If you want to automatically center the text horizontally and vertically in the merged cells, set the [MergeAdjustView](#) property to True.

Working with Groups

For an overview about groups and how they work, see [Groups](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating Groups](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Referencing a Group](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Applying Properties to a Specific Group](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating Children of Groups](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Calculating the Width of Group Children](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Providing Group Headers](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Resizing Groups](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Customizing Groups](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Moving Groups in the Control](#)

Creating Groups

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can group columns in a multiple-column fpCombo or fpList control. You can create any number of groups. Groups are zero-based and are numbered from left to right and top to bottom. Once you have created groups, you can reference specific groups and customize them, as explained in the following topics.

Tip If you are using groups, for best results assign all columns to groups.

Groups can be children of other groups. For information on how to make a group a child of another group, see [CreatingChildren of Groups](#).

[Print](#)

[Copy](#)

[Close](#)

To create groups

Designer Page

1. On the [Specific subtab of the Groups designer page](#), select the [Groups](#) property from the properties list.
2. Type the number of groups in the box under Groups in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To create groups

Browser/Code

Specify the number of groups with the [Groups](#) property.

Referencing a Group

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

List Pro controls have three methods to reference a specific group: index number, name, and identifier number.

Group index numbers are assigned by the control and based on the physical position of the group in the control. These numbers are zero-based, and groups are numbered from left to right and top to bottom within their parent group, if any, and then within the control. If you move or insert a group, the control changes the index numbers. Use the [Grp](#) property to reference groups by their index numbers.

You can also define a unique name or identifier number for each group. If you use a group name or identifier number to specify a group, the properties you set for a particular group are applied to that group, regardless of its position in the control. Use the [GrpName](#) property to define group names and the [GrpID](#) property to define group identifier numbers. Typically, you should use either the name or the identifier number, but not both.

For example, assume you define four groups in your control as follows.

Grp	GrpID	GrpName
0	1	Monday
1	2	Tuesday
2	3	Wednesday
3	4	Thursday

The control would appear as shown below.

Grp = 0, GrpID = 1, GrpName = Monday	Grp = 1, GrpID = 2, GrpName = Tuesday	Grp = 2, GrpID = 3, GrpName = Wednesday	Grp = 3, GrpID = 4, GrpName = Thursday
---	--	--	---

As the first example, assume you move the third group (Grp = 2) to the far left side of the control. The control now appears as shown below.

Grp = 0, GrpID = 3, GrpName = Wednesday	Grp = 1, GrpID = 1, GrpName = Monday	Grp = 2, GrpID = 2, GrpName = Tuesday	Grp = 3, GrpID = 4, GrpName = Thursday
--	---	--	---

Note that the group index numbers changed but the group identification numbers and group names did not change.

As the second example, assume you make the third and fourth groups children of the first group (Grp = 0). The control now appears as shown below.

Grp = 0, GrpID = 1, GrpName = Monday		Grp = 3, GrpID = 2, GrpName = Tuesday
Grp = 1, GrpID = 3, GrpName = Wednesday	Grp = 2, GrpID = 4, GrpName = Thursday	

Note that the group index numbers changed and are numbered from left to right and top to bottom within their parent group.

Once you have specified the number of groups in the control and created identifier numbers or names, you can specify custom features for each group individually. For more information on setting properties for specified groups, see [Applying Properties to a Specific Group](#).

[Print](#)

[Copy](#)

[Close](#)

To reference a group

Designer Page

1. Specify the group. On the [Specific subtab of the Groups designer page](#), under Individual, select the group number from the Grp drop-down list box.
2. If you want to define a group identifier number,
 - a. Select the [GrpID](#) property from the properties list box.
 - b. Type the identifier number in the box under GrpID in the property value area.
3. If you want to define a group name,
 - a. Select the [GrpName](#) property from the properties list box.
 - b. Type the name in the box under GrpName in the property value area.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

4. Repeat steps 1

✓	✓	✓	✓	✓
---	---	---	---	---

 3 until all groups have been defined.
5. To specify the group to which an action or property applies, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.

[Print](#)

[Copy](#)

[Close](#)

To reference a group

Code

1. Specify the group with the [Grp](#) property.
2. If you want to define a group identification number, set the [GrpID](#) property.
3. If you want to define a group name, set the [GrpName](#) property.
4. Specify the group to which an action or property applies with the Grp, [GrpFromID](#), or [GrpFromName](#) property.

Applying Properties to a Specific Group

Once you have specified the number of groups in the control and defined the group identifier number or name, you can use the designated-group properties to perform an action on a specific group. Designated-group properties require that the [Grp](#), [GrpFromID](#), or [GrpFromName](#) properties be set before using them.

The following table contains designated-group properties for the List Pro controls.

GrpHeaderText	GrpPos
GrpHide	GrpPosInParent
GrpLockResize	GrpWidth
GrpParentGroup	

If you want to use the designated-group properties in a List Pro control with only one group, you must set the [Groups](#) property to 1 and the Grp property to 0.

Creating Children of Groups

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

You can group columns or other groups together. For example, you might want to make all the columns related to a clients address, such as street information, city, state, and zip code, children of an address group.

A child of a group can be another group or a column, but not a combination of the two. For example, if Group 2 has two children, those children can be either two other groups or two columns; they cannot be one group and one column.

Children of groups exhibit the following characteristics:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If you move a group, the groups children move with it. For more specific information, see [Moving](#)

[Groups in the Control](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

When you hide a group, the groups children are also hidden.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Children are automatically sized to fit the group width. This can result in text not being fully displayed in a column. For more information, see [Calculating the Width of Group Children](#).

You can create children of groups by either

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Using the [ColParentGroup](#) or [GrpParentGroup](#) properties

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Dragging and dropping columns or groups under their new parent group

For more information on moving columns and groups, see [Using the Drag-and-Drop Method to Move Columns](#) and [Using the Drag-and-Drop Method to Move Groups](#).

[Print](#)

[Copy](#)

[Close](#)

To create children of groups

Designer Page

1. If you want to make a column a child of a group,
 - a. Specify the column. On the [Specific subtab of the Columns designer page](#), either select the column from the Col, Col Name, or Col ID drop-down list box or click the column in the preview area.
 - b. Specify the parent group.
 - i. Select the [ColParentGroup](#) property from the properties list box.
 - ii. Type the parent groups index number in the ColParentGroup box in the property value area.
2. If you want to make another group the child of a group,
 - a. Specify the group. On the [Specific subtab of the Groups designer page](#), either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
 - b. Specify the parent group.
 - i. Select the [GrpParentGroup](#) property from the properties list box.
 - ii. Type the parent groups index number in the GrpParentGroup box in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To create children of groups

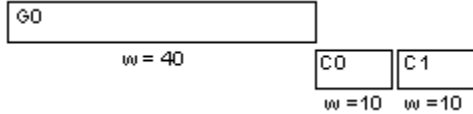
Code

1. If you want to make a column a child of a group,
 - a. Specify the column with the [Col](#), [ColFromID](#), or [ColFromName](#) property.
 - b. Specify the parent group of the column with the [ColParentGroup](#) property.
2. If you want to make another group the child of a group,
 - a. Specify the group with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.
 - b. Specify the parent group of the group with the [GrpParentGroup](#) property.

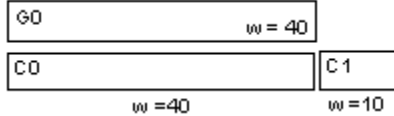
Calculating the Width of Group Children

If children of groups exist, widths for group children on any given level are adjusted to fit within the parent group width. If you have one child in one group, the child width is set equal to the group width, regardless of the [ColWidth](#) property setting. As you add additional children to the group, the child widths are adjusted proportionally to their current widths.

For example, assume you define two columns and one group.



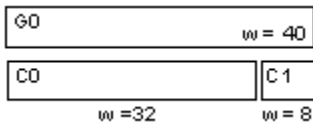
When you make the first column (C0) a child of the group (G0), its column width is set to the width of the group (40).



When you make the second column (C1) a child of the group, the width of each column is recalculated according to the following formula.

$$\text{New column width} = \frac{\text{Current column width}}{\sum \text{Current widths of columns to be under the group}} \times \text{Group width}$$

The new width of column C0 is 32 $[40/(40+10) \cdot 40]$. The new width of column C1 is 8 $[10/(40+10) \cdot 40]$. The control now appears as follows.



After columns are assigned to groups, you can resize them with the mouse or by specifying the column width. For more information, see [Resizing Columns](#).

Providing Group Headers

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating Group Headers](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Customizing Group Headers](#)

Creating Group Headers

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

By default, group headers are not displayed. You can display group headers for List Pro controls. The header contains static text that does not scroll vertically with the list. Headers can be useful for labeling multiple-group lists.

Also by default, the group header height is automatically sized to fit the text. You can define a specific header height.

You can specify the text that appears in the group headers. You can also customize the text. For more information about customizing the text, see [Working with Text and Graphics](#).

[Print](#)

[Copy](#)

[Close](#)

To create group headers

Designer Page

1. Specify the group. On the [Specific subtab of the Groups designer page](#), either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
2. Specify the header text for each group.
 - a. Select the [GrpHeaderText](#) property from the properties list box.
 - b. Type the header text in the box under GrpHeaderText in the property value area.
3. Repeat steps 1 and 2 until you have specified the header text for all the group headers.
4. Display the group headers.
 - a. On the [General subtab of the Groups designer page](#), select the [GroupHeaderShow](#) property in the properties list.
 - b. Select the True option button under GroupHeaderShow in the property value area.
5. Specify the group header height.
 - a. Select the [GroupHeaderHeight](#) property in the properties list.
 - b. Type the header height in twips in the box under GroupHeaderHeight.

If the header text is long, you can wrap the text to multiple lines. For more information, see [Wrapping Text in aList Pro Control](#).

[Print](#)

[Copy](#)

[Close](#)

To create group headers

Browser/Code

1. To specify the group header text, at run time,
 - a. Specify the group with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.
 - b. Specify the group header text with the [GrpHeaderText](#) property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

- c. Repeat steps 2.a
2. Set the [GroupHeaderShow](#) property to True to display the headers.
3. To specify the group header height, set the [GroupHeaderHeight](#) property.

If the header text is long, you can wrap the text to multiple lines. For more information, see [Wrapping Text in a List Pro Control](#).

Customizing Group Headers

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

You can customize the way group headers are displayed in an fpCombo or fpList control. You can customize characteristics such as displaying and aligning pictures, changing text color and appearance, displaying lines, and changing the background color.

[Print](#)

[Copy](#)

[Close](#)

To customize group headers

Designer Page

1. Specify which header or headers to customize.
 - a. On the [List subtab of the ApplyTo designer page](#), choose either 8 - Grp Headers or 10 - Single Grp Header from the List Apply To drop-down list box.
 - b. If you are customizing an individual group header, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area to specify the group.
2. Follow the instructions in the appropriate topic listed in the following table.

For more information on . . .

Pictures and picture alignment

Text color, appearance, and alignment

Lines

Background color

See . . .

[Working with Text and Graphics](#)

[Working with Text and Graphics](#)

[Customizing Lines](#)

[Changing the Background Color](#)

[Print](#)

[Copy](#)

[Close](#)

To customize group headers

Code

1. Set the [ListApplyTo](#) property to 8 (Group Headers) or 10 (Individual Group Header).
2. If you are customizing an individual group header, specify the group with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.
3. Follow the instructions in the appropriate topic listed in the following table.

For more information on . . .

Pictures and picture alignment

Text color, appearance, and alignment

Lines

Background color

See . . .

[Working with Text and Graphics](#)

[Working with Text and Graphics](#)

[Customizing Lines](#)

[Changing the Background Color](#)

Resizing Groups

You can use either of the following methods to change the width of a group:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Use the mouse](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Specify a group width](#)

Using the Mouse to Resize Groups

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

By default, you cannot resize groups using the mouse. Using the [AllowGrpResize](#) property, you can resize any unlocked group

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

by dragging the group or header boundary to a new position with the resize pointer. For more information about locking groups against resizing, see [Customizing Groups](#).

[Print](#)

[Copy](#)

[Close](#)

To resize groups using the mouse

Designer Page

1. Specify how groups are resized.
 - a. On the [General subtab of the Groups designer page](#), select the [AllowGrpResize](#) property from the properties list box.
 - b. Select either the 1 - Resize Header or 2 - Resize Grp or Header option button in the property value area.
2. In the preview area, position the mouse pointer over the group boundary (if you chose option button 2 - Resize Grp or Header) or header boundary of the group you want to move.
3. When the mouse pointer changes to the resize pointer, press the left mouse button and drag the boundary to resize the group.

[Print](#)

[Copy](#)

[Close](#)

To resize groups using the mouse

Browser/Code

1. Set the [AllowGrpResize](#) property to either 1 (Resize Header) or 2 (Resize Grp or Header).
2. At run time,
 - a. Position the mouse pointer over the group boundary (if you set the AllowGrpResize property to 2 (Resize Grp or Header)) or header boundary of the group you want to move.
 - b. When the mouse pointer changes to the resize pointer, press the left mouse button and drag the boundary to resize the group.

Specifying the Group Width

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can set the width of a group in a List Pro control to a specific value. The default unit of measurement for group width is the average character width of the current font. However, you can specify the width in twips, pixels, or the maximum character width of the current font.

Note If groups are children of other groups, the child group width is determined by the width of the parent group and other group children. For more information, see [Calculating the Width of Group Children](#). If you want to set the width of a group that is a child of another group, use the following procedure after making the group a child of another group.

[Print](#)

[Copy](#)

[Close](#)

To specify the group width

Designer Page

1. If you want to specify a different unit of measurement,
 - a. On the [General subtab of the Columns designer page](#), select the [ColumnWidthScale](#) property from the properties list box.
 - b. Select the appropriate option button under ColumnWidthScale in the property value area.
2. Specify the group. On the [Specific subtab of the Groups designer page](#), either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
3. Specify the group width.
 - a. Select the [GrpWidth](#) property from the properties list box.
 - b. Type the width in the box under GrpWidth in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To specify the group width

Browser/Code

1. If you want to specify a different unit of measurement, set the [ColumnWidthScale](#) property.
2. At run time,
 - a. Specify the group with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.
 - b. Set the [GrpWidth](#) property.

Customizing Groups

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can customize the groups of a List Pro control. You can

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Hide a group in the list

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Freeze a specific number of leftmost groups from scrolling horizontally

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Prevent a specific group from being resized using the mouse

You can also customize the following group attributes.

For more information on . . .

Pictures and picture alignment

Text color, appearance, and alignment

Lines

Background color

See . . .

[Working with Text and Graphics](#)

[Working with Text and Graphics](#)

[Customizing Lines](#)

[Changing the Background Color](#)

[Print](#)

[Copy](#)

[Close](#)

To customize groups

Designer Page

1. If you want to hide a group in the list,
 - a. Specify the group. On the [Specific subtab of the Groups designer page](#), either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
 - b. Hide the group.
 - i. Select the [GrpHide](#) property from the properties list.
 - ii. Select the True option button under GrpHide in the property value area.

Note When you hide a group, the children of the group are also hidden.
2. If you want to prevent a set number of leftmost groups from scrolling horizontally,
 - a. On the [General subtab of the Groups designer page](#), select the [GrpsFrozen](#) property from the properties list box.
 - b. Type the number of leftmost groups to remain frozen in the box under GrpsFrozen in the property value area.
3. If you want to prevent a group from being resized using the mouse,
 - a. Specify the group. On the Specific subtab of the Groups designer page, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
 - b. Lock the group against resizing.
 - i. Select the [GrpLockResize](#) property from the properties list box.
 - ii. Select the True option button from the property value area.

[Print](#)

[Copy](#)

[Close](#)

To customize groups

Browser/Code

1. If you want to hide a group, at run time,
 - a. Specify the group you want to hide with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.
 - b. Set the [GrpHide](#) property to True.
Note When you hide a group, the children of the group are also hidden.
2. If you want to prevent a set number of leftmost groups from scrolling horizontally, set the [GrpsFrozen](#) property to the number of leftmost groups to remain frozen.
3. If you want to prevent a specific group from being resized using the mouse, at run time,
 - a. Specify the group you want to prevent from being resized with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.
 - b. Set the [GrpLockResize](#) property to True.

Moving Groups in the Control

You can use either of the following methods to move or position groups in a control:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Use the drag-and-drop method](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Specify the position of the group](#) using the [GrpPos](#) or [GrpPosInParent](#) property

Using the Drag-and-Drop Method to Move Groups

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can allow movement of any group or any nonfrozen group using the drag-and-drop method, assuming the control displays group headers. For more information on freezing groups, see [Resizing Groups](#). For more information on displaying group headers, see [Creating Group Headers](#).

When you move a group to the left of its current position, all groups to the right of the destination group up to the original position of the moved group will shift to the right. When you move a group to the right of its current position, all groups between the original position of the moved group and its destination position will shift to the left. When you move a group, the index number of any group affected by the move will change.

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Because group index numbers are based on the physical position of the group in the control, if you plan on moving groups, use one of the unique group identifiers ([GrpID](#) or [GrpName](#) property) to reference a group rather than the [Grp](#) property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

You must display group headers to move groups using the drag-and-drop method.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If you move a group, the groups children move with it.

For example, assume you have the following list box.

Group 0	Group 1	Group 2	Group 3
Col 0	Col 1	Col 2	Col 3
A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T

As the first example, if you move Group 3 to the left of Group 1, both Group 1 and Group 2 are shifted to the right as shown in the following figure.

Group 0	Group 3	Group 1	Group 2
Col 0	Col 3	Col 1	Col 2
A	D	B	C
E	H	F	G
I	L	J	K
M	P	N	O
Q	T	R	S

Starting with the original layout, if you move Group 0 to the right of Group 2, both Group 1 and Group 2 are shifted to the left as shown in the following figure.

Group 1	Group 2	Group 0	Group 3
Col 1	Col 2	Col 0	Col 3
B	C	A	D
F	G	E	H
J	K	I	L
N	O	M	P
R	S	Q	T

You can make a group a child of another group by dragging and dropping the group under another group. The moved groups children move with the group and are displayed to the left of the destination groups children. The destination groups children

become children of the child group.

For example, assume you have the following list box.

Group 0	Group 1	Group 2	Group 3
Col 0	Col 1	Col 2	Col 3
A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T

If you move Group 2 under Group 1, Col 2 (child of Group 2) is displayed to the left of Col 1 (child of Group 1). Col 1 is now a child of Group 2.

Group 0	Group 1		Group 3
	Group 2		
Col 0	Col 2	Col 1	Col 3
A	C	B	D
E	G	F	H
I	K	J	L
M	O	N	P
Q	S	R	T
U	W	V	X

Suppose you also want to make Group 3 a child of Group 1. If you move Group 3 under Group 1, Group 3 and its child (Col 3) are displayed to the left of Group 2 (child of Group 1).

Group 0	Group 1		
	Group 3	Group 2	
Col 0	Col 3	Col 2	Col 1
A	D	C	B
E	H	G	F
I	L	K	J
M	P	O	N
Q	T	S	R
U	X	W	V

Instead, suppose you want to make Group 3 a child of Group 2. If you move Group 3 under Group 2, Col 3 (child of Group 3) is displayed to the left of Col 1 and Col 2 (children of Group 1). Col 1 and Col 2 are now children of Group 3.

Group 0	Group 1		
	Group 2		
	Group 3		
Col 0	Col 3	Col 2	Col 1
A	D	C	B
E	H	G	F
I	L	K	J
M	P	O	N
Q	T	S	R
U	X	W	V

Note You may want to resize the columns and groups after moving a group under another group. For instructions, see [Resizing Groups](#).

[Print](#)

[Copy](#)

[Close](#)

To drag and drop groups

Designer Page

1. Specify which groups you want to drag and drop.
 - a. On the [General subtab of the Groups property page](#), select the [AllowGrpDragDrop](#) property from the properties list box.
 - b. Select either the 1- All Grps or 2 -Non Frozen Grps option button under AllowGrpDragDrop in the property value area.
2. In the preview area, position the mouse pointer over the header of the group you want to move.
3. When the mouse pointer changes to the drag-drop pointer (a hand), press the left mouse button and drag and drop the group in its new location.

[Print](#)

[Copy](#)

[Close](#)

To drag and drop groups

Browser/Code

1. Set the [AllowGrpDragDrop](#) property to either 1 (All Grps) or 2 (Non Frozen Grps).
2. At run time,
 - a. Move the mouse pointer over the header of the group you want to move.
 - b. When the mouse pointer changes to the drag-drop pointer (a hand), press the left mouse button and drag and drop the group in its new location.

Defining the Position of Groups in the Control

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

You can move groups by defining their position within a List Pro control. If a group is a child of a parent group, you can also define the position of that group within its parent group.

Position numbers are zero-based and are numbered from left to right and top to bottom within a parent group, if any, and then within the control.

Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Because group index numbers are based on the physical position of the group in the control, if you plan on moving groups, use one of the unique group identifiers ([GrpID](#) or [GrpName](#) property) to reference a group rather than the [Grp](#) property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Define all groups ([Grp](#), [GrpID](#), and [GrpName](#) properties) in the control before moving groups with the [GrpPos](#) or [GrpPosInParent](#) property.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If you have groups, you should use only the [GrpPosInParent](#) property to position groups within other groups. If you use the [GrpPos](#) property and the [GrpPosInParent](#) property at the same time, you might get unpredictable results.

[Print](#)

[Copy](#)

[Close](#)

To position a group in the control

Designer Page

1. Specify the group. On the [Specific subtab of the Groups designer page](#), either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
2. If you want to specify the position of the group in the control,
 - a. Select the [GrpPos](#) property in the properties list box.
 - b. Type the position number in the box under GrpPos in the property value area.
3. If you want to specify the position of the group in its parent group,
 - a. Select the [GrpPosInParent](#) property in the properties list box.
 - b. Type the position number in the box under GrpPosInParent in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To position a group in the control

Code

1. Specify the group with the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.
2. If you want to specify the position of the group in the control, set the [GrpPos](#) property.
3. If you want to specify the position of the group in its parent group, set the [GrpPosInParent](#) property.

Customizing the Controls Appearance

For an overview about the controls appearance, see [Text and Graphics](#) and [Appearance](#).

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Working with Text and Graphics](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Working with Lines](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Changing the Background Color](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Changing the List Gray Area Color](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Displaying and Customizing Scroll Bars](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Specifying the Number of Rows Displayed in the Drop-Down List](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Displaying Drop Shadows](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Customizing the Controls Borders](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating a Frame](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Displaying Focus on the Control](#)

Working with Text and Graphics

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Overview](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Displaying Graphics](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Aligning Text and Graphics](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Orienting Text and Graphics](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Wrapping Text in a List Pro Control](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Changing Text Color and Fonts](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Creating Three-Dimensional Text](#)

Overview

A List Pro control can contain text, graphics, or both. You can align text horizontally or vertically within cells, rows, columns, and headers. If the control contains both text and graphics, they can be aligned separately or in relation to one another.

If you have text in a List Pro control, you can specify whether the text displays on one line or wraps and displays on multiple lines. Text can be displayed horizontally or vertically. Text and graphics can be rotated 90, 180, or 270 degrees from the horizontal position.

You can change the look of text by creating a three-dimensional effect. You can also change the text color.

Displaying Graphics

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Code Instructions](#)

Overview

An fpCombo or fpList control can contain text, graphics, or both. You can also display a different graphic when an item or row of items is selected.

If the control contains both text and graphics, they can be aligned separately or in relation to one another. For more information on aligning graphics, see [Aligning Text and Graphics](#). Text and graphics can also be rotated 90, 180, or 270 degrees from the horizontal position. For more information, see [Orienting Text and Graphics](#).

Use the [ListApplyTo](#) property to specify where graphics are displayed. For more information on using the ListApplyTo property, see [Applying Properties to Specific Parts of the Control](#).

[Print](#)

[Copy](#)

[Close](#)

To display graphics

Designer Page

1. Select where you want to display the graphics.
 - a. On the [List subtab of the ApplyTo designer page](#), select an item from the List Apply To drop-down list box.
 - b. If you chose 9 - Single Col Header or 12 - Single Item to display graphics in a specific cell, column, column header, or row, either select the column from either the Col, Col Name, or Col ID drop-down list box and type the row in the Row box as appropriate, or click the cell, column, or row in the preview area.
 - c. If you chose 10 - Single Group Header or 11 - Single Group to display graphics in a specific group or group header, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
2. Specify the graphic to be displayed.
 - a. Select the [Picture](#) property from the properties list box.
 - b. Choose the Picture button under Picture in the property value area to display the file selection dialog box.
 - c. Specify the path and filename and choose the OK button.

The selected graphics file is displayed in the picture preview area.
3. If you want a different graphic displayed when a list item is selected, repeat step 2 but select the [PictureSel](#) property in step 2.a.

[Print](#)

[Copy](#)

[Close](#)

To display graphics

Code

1. Specify where you want to display the graphics.
 - a. Set the [ListApplyTo](#) property.
 - b. If you want to display the graphics in a specific cell, column, column header, or row, set the [Col](#), [ColFromID](#), or [ColFromName](#) property and the [Row](#) property as appropriate.
 - c. If you want to display the graphics in a specific group or group header, set the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.
2. Specify the graphic to be displayed with the [Picture](#) property.
3. If you want a different graphic displayed when a list item is selected, specify the graphic with the [PictureSel](#) property.

Aligning Text and Graphics

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can set the horizontal alignment of the controls contents to display either to the left, center, or right within a cell or header. You can set the vertical alignment of the controls contents to display either to the top, center, or bottom within a cell or header.

In the following figure, the text and graphic are horizontally centered in the cell. The graphic is vertically aligned below the text.



Use the [ListApplyTo](#) property to specify where to apply the alignment. Alignment can be applied to any of the areas described in [Applying Properties to Specific Parts of the Control](#).

[Print](#)

[Copy](#)

[Close](#)

To align text and graphics

Designer Page

1. If you want to specify where the alignment is applied,
 - a. On the [List subtab of the ApplyTo designer page](#), select an item from the List Apply To drop-down list box.
 - b. If you chose 9 - Single Col Header or 12 - Single Item to align text and graphics in a specific cell, column, column header, or row, either select the column from either the Col, Col Name, or Col ID drop-down list box and type the row in the Row box as appropriate, or click the cell, column, or row in the preview area.
 - c. If you chose 10 - Single Group Header or 11 - Single Group to align text and graphics in a specific group or group header, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
2. If you want to specify how the text should be horizontally or vertically aligned,
 - a. On the List subtab of the ApplyTo designer page, select the [AlignH](#) or [AlignV](#) property from the properties list box.
 - b. Select the appropriate option button under AlignH or AlignV in the property value area.
3. If you want to specify how the graphic should be horizontally or vertically aligned,
 - a. On the List subtab of the ApplyTo designer page, select the [PictureAlignH](#) or [PictureAlignV](#) property from the properties list box.
 - b. Select the appropriate list item from the drop-down combo box under PictureAlignH or PictureAlignV in the property value area.

Note If you are displaying text and graphics within the fpCombo or fpList control, set the PictureAlignH property to 4 (Left of Text) or 5 (Right of Text) and the PictureAlignV property to 4 (Top of Text) or 5 (Bottom of Text) for best results.

[Print](#)

[Copy](#)

[Close](#)

To align text and graphics

Browser/Code

1. If you want to specify where the alignment is applied, at run time,
 - a. Set the [ListApplyTo](#) property.
 - b. If you want to apply the alignment to a specific cell, column, column header, or row, set the [Col](#), [ColFromID](#), or [ColFromName](#) property and the [Row](#) property as appropriate.
 - c. If you want to apply the alignment to a group or group header, set the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Note If you specified where the alignment is applied in step 1, you should also perform steps 2 and 3 at run time.

2. If you want to specify how the text should be horizontally and vertically aligned, set the [AlignH](#) and [AlignV](#) properties, respectively.
3. If you want to specify how the graphic should be horizontally and vertically aligned, set the [PictureAlignH](#) and [PictureAlignV](#) properties, respectively.

Note If you are displaying text and graphics within the fpCombo or fpList control, set the PictureAlignH property to 4 (Left of Text) or 5 (Right of Text) and the PictureAlignV property to 4 (Top of Text) or 5 (Bottom of Text) for best results.

Orienting Text and Graphics

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can display text in a List Pro control horizontally or vertically. You can also rotate text and graphics 90, 180, or 270 degrees from the horizontal position.

In the following figure the text and graphics are rotated 270 degrees from the horizontal position.



Notes

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If the text orientation is vertical, you can specify that text wraps from right to left or left to right.

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

If the font you choose cannot be rotated, List Pro substitutes a similar, TrueType® font that can be rotated.

Use the [ListApplyTo](#) property to specify where to apply text and graphic orientation. Text and graphic orientation can be applied to any of the areas described in [Applying Properties to Specific Parts of the Control](#).

[Print](#)

[Copy](#)

[Close](#)

To set text and graphic orientation

Designer Page

1. If you want to specify where the orientation is applied,
 - a. On the [List subtab of the ApplyTo designer page](#), select an item from the List Apply To drop-down list box.
 - b. If you chose 9 - Single Col Header or 12 - Single Item to orient text and graphics in a specific cell, column, column header, or row, either select the column from either the Col, Col Name, or Col ID drop-down list box and type the row in the Row box as appropriate, or click the cell, column, or row in the preview area.
 - c. If you chose 10 - Single Group Header or 11 - Single Group to orient text and graphics in a specific group or group header, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
2. Specify the text orientation.
 - a. On the List subtab of the ApplyTo designer page, select the [TextOrientation](#) property from the properties list box.
 - b. Select the appropriate list item from the drop-down combo box under TextOrientation in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To set text and graphic orientation

Browser/Code

1. If you want to specify where the orientation is applied, at run time,
 - a. Set the [ListApplyTo](#) property.
 - b. If you want to apply the orientation to a specific cell, column, column header, or row, set the [Col](#), [ColFromID](#), or [ColFromName](#) property and the Row property as appropriate.
 - c. If you want to apply the orientation to a group or group header, set the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Note If you specified where the orientation is applied in step 1, you should also perform step 2 at run time.

2. Set the [TextOrientation](#) property.

Wrapping Text in a List Pro Control

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

By default, text displays on a single line. If the column or group width does not allow the text to fit on one line, you can specify that the text wrap to the next line. You can adjust the row height to accommodate multiple lines of text.

You can specify the maximum width of a column or group in a List Pro control. For information on changing the width of a column, see [Customizing Columns](#). For information on changing the width of a group, see [Resizing Groups](#).

Use the [ListApplyTo](#) property to specify where to apply text wrap. Text can wrap in any of the areas described in [Applying Properties to Specific Parts of the Control](#).

[Print](#)

[Copy](#)

[Close](#)

To let text wrap

Designer Page

1. If you want to specify where text wraps,
 - a. On the [List subtab of the ApplyTo designer page](#), select an item from the List Apply To drop-down list box.
 - b. If you chose 9 - Single Col Header or 12 - Single Item to wrap text in a specific cell, column, column header, or row, either select the column from either the Col, Col Name, or Col ID drop-down list box and type the row in the Row box as appropriate, or click the cell, column, or header in the preview area.
 - c. If you chose 10 - Single Group Header or 11 - Single Group to wrap text in a specific group or group header, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
2. Turn on text wrapping.
 - a. On the List subtab of the ApplyTo designer page, select the [MultiLine](#) property from the properties list box.
 - b. Select the 2 - Multiple Line option button under MultiLine in the property value area.
3. If you need to adjust the row height,
 - a. On the [Misc subtab of the Appearance designer page](#), type the row number in the Row box.
 - b. Select the [RowHeight](#) property in the properties list box.
 - c. Type the row height in twips in the box under RowHeight in the property value area.
4. If you need to adjust the column header height,
 - a. On the [General subtab of the Columns designer page](#), select the [ColumnHeaderHeight](#) property in the properties list box.
 - b. Type the column header height in twips in the box under ColumnHeaderHeight in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To let text wrap

Browser/Code

1. If you want to specify where text wraps, at run time,
 - a. Set the [ListApplyTo](#) property.
 - b. If you want to apply the text wrap to a specific cell, column, column header, or row, set the [Col](#), [ColFromID](#), or [ColFromName](#) property and the [Row](#) property as appropriate.
 - c. If you want to apply the text wrap to a specific group or group header, set the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Note If you specified where text wrap is applied in step 1, you should also perform step 2 at run time.

2. To turn on text wrapping, set the [MultiLine](#) property to 2 (Multiple Line).
3. If you need to adjust the row height, specify the row height in twips with the [RowHeight](#) property.
4. If you need to adjust the column header height, set the column header height in twips with the [ColumnHeaderHeight](#) property.

Changing Text Color and Fonts

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can change the color and font characteristics of text in a List Pro control.

If you create three-dimensional text, you can specify highlight and shadow colors. For more information on three-dimensional text, see [Creating Three-Dimensional Text](#).

Use the [ListApplyTo](#) property to specify where to apply text color and fonts. Text color and fonts can be applied to any of the areas described in [Applying Properties to Specific Parts of the Control](#).

[Print](#)

[Copy](#)

[Close](#)

To change text color and fonts

Designer Page

1. If you want to specify where the text color or font characteristics are applied,
 - a. On the [List subtab of the ApplyTo designer page](#), select an item from the List Apply To drop-down list box.
 - b. If you chose 9 - Single Col Header or 12 - Single Item to apply the color or font characteristics in a specific cell, column, column header, or row, either select the column from either the Col, Col Name, or Col ID drop-down list box and type the row in the Row box as appropriate, or click the cell, column, or row in the preview area.
 - c. If you chose 10 - Single Group Header or 11 - Single Group to apply the color or font characteristics in a specific group or group header, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
2. If you want to change the text color,
 - a. On the List subtab of the ApplyTo designer page, select the [ForeColor](#) property from the properties list box.
 - b. Choose the Color button under ForeColor in the property value area to display the ForeColor dialog box.
 - c. Select a basic color or define your own custom color.
 - d. Choose the OK button.
3. If you want to customize the font characteristics,
 - a. On the List subtab of the ApplyTo designer page, select the [Font](#) property from the properties list box.
 - b. Choose the Font button under Font in the property value area to display the Font dialog box.
 - c. Select the font characteristics, and then choose the OK button.
4. If you want to clear the font characteristics for the area and inherit the characteristics of the areas hierarchical predecessor (see Appendix C, "Hierarchy of Property Settings" in the printed *List Pro User's Guide*),
 - a. On the List subtab of the ApplyTo designer page, select the [FontEmpty](#) property from the properties list box.
 - b. Select the True option button under FontEmpty in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To change text color and fonts

Browser/Code

1. If you want to specify where the text color or font characteristics are applied, at run time,
 - a. Set the [ListApplyTo](#) property.
 - b. If you want to apply the color or font characteristics to a specific cell, column, column header, or row, set the [Col](#), [ColFromID](#), or [ColFromName](#) property and the [Row](#) property as appropriate.
 - c. If you want to apply the color or font characteristics to a group or group header, set the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Note If you specified where the text color or font characteristics are applied in step 1, you should also perform steps 2

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

4 at run time.

2. If you want to change the text color, set the [ForeColor](#) property.
3. If you want to customize the font characteristics, specify the characteristics with the [Font](#) property or specific font properties such as [FontBold](#), [FontName](#), or [FontSize](#).
4. If you want to clear the font characteristics for the area and inherit the characteristics of the areas hierarchical predecessor, set the [FontEmpty](#) property to True.

For more information on hierarchical predecessors, see Appendix C, "Hierarchy of Property Settings" in the printed *List Pro User's Guide*.

Creating Three-Dimensional Text

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can apply three-dimensional effects to a List Pro controls text. You can also specify the text offset (the distance between the text and the highlight and shadow colors).

Use the [ListApplyTo](#) property to specify where to apply three-dimensional text effects. Three-dimensional text can be displayed in any of the areas described in [Applying Properties to Specific Parts of the Control](#).

[Print](#)

[Copy](#)

[Close](#)

To create three-dimensional text

Designer Page

1. If you want to specify where three-dimensional effects are applied,
 - a. On the [List subtab of the ApplyTo designer page](#), select an item from the List Apply To drop-down list box.
 - b. If you chose 9 - Single Col Header or 12 - Single Item to apply the three-dimensional text characteristics in a specific cell, column, column header, or row, either select the column from either the Col, Col Name, or Col ID drop-down list box and type the row in the Row box as appropriate, or click the cell, column, or row in the preview area.
 - c. If you chose 10 - Single Group Header or 11 - Single Group to apply the three-dimensional text characteristics in a specific group or group header, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
2. Turn on three-dimensional text effects.
 - a. On the List subtab of the ApplyTo designer page, select the [List3DText](#) property from the properties list box.
 - b. Select an item other than 0 - None from the drop-down combo box under List3DText in the property value area.
3. If you want to customize the text offset,
 - a. Select the [List3DTextOffset](#) property from the properties list box.
 - b. Type the offset in pixels in the box under List3DTextOffset in the property value area.
4. If you want to customize the highlight or shadow color,
 - a. Select the [List3DTextHighlightColor](#) or [List3DTextShadowColor](#) property from the properties list box.
 - b. Choose the Color button under List3DTextHighlightColor or List3DTextShadowColor in the property value area to display the List3DTextHighlightColor or List3DTextShadowColor dialog box.
 - c. Select a basic color or define your own custom color.
 - d. Choose the OK button.

[Print](#)

[Copy](#)

[Close](#)

To create three-dimensional text

Browser/Code

1. If you want to specify where three-dimensional effects are applied,
 - a. Specify where you want the three-dimensional effects applied with the [ListApplyTo](#) property.
 - b. If you want to apply the three-dimensional effects to a specific cell, column, column header, or row, set the [Col](#), [ColFromID](#), or [ColFromName](#) property and the [Row](#) property as appropriate.
 - c. If you want to apply the three-dimensional effects to a specific group or group header, set the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Note If you specified where the three-dimensional effects are applied in step 1, you should also perform steps 2

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

4 at run time.

2. Set the [List3DText](#) property to a value other than 0 (None) to turn on three-dimensional effects.
3. Specify the text offset in pixels with the [List3DTextOffset](#) property.
4. Specify the highlight and shadow colors with the [List3DTextHighlightColor](#) and [List3DTextShadowColor](#) properties.

Working with Lines

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Overview](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Applying Properties to Specific Lines](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

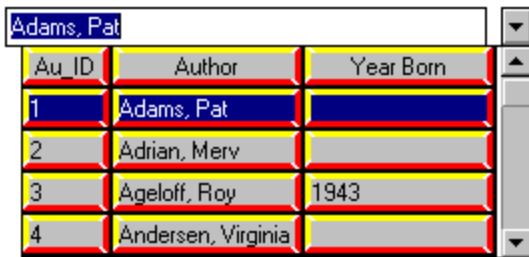
[Adding Lines](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Customizing Lines](#)

Overview

You can display lines in the control and these lines can be flat, three-dimensional, or both. The following combo box displays both flat and three-dimensional lines.



Au_ID	Author	Year Born
1	Adams, Pat	
2	Adrian, Merv	
3	Ageloff, Roy	1943
4	Andersen, Virginia	

You can specify where lines are displayed with the [ListApplyTo](#) property, the [LineApplyTo](#) property, or both. Lines can be displayed in any of the areas described in [Applying Properties to Specific Parts of the Control](#) and [Applying Properties to Specific Lines](#).

You can specify how the lines look, first, by specifying which lines are affected (see [Applying Properties to Specific Lines](#)), and second, by specifying what characteristics to change (see [Customizing Lines](#)).

Applying Properties to Specific Lines

You can use the following designated-line properties to customize the appearance of lines in a List Pro control:

[Line3DDark](#)

[LineColor](#)

[Line3DLight](#)

[LineStyle](#)

[Line3DWidth](#)

[LineWidth](#)

As described in the [Overview](#), you can use the [ListApplyTo](#) property, the [LineApplyTo](#) property, or both to specify where the designated-line properties apply. You can use the [LineApplyTo](#) property to apply these properties to all lines, the lines between rows, the lines between columns, or the vertical lines between columns when multiple levels exist. By default, designated-line properties apply to all lines.

Adding Lines

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

By default, neither horizontal nor vertical lines are displayed in a List Pro control. You can display flat lines, three-dimensional lines, or both. These lines can be displayed between rows, between columns, between columns when multiple levels exist, or between all these areas together.

The following figures illustrate the lines and their appearance:

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Flat lines

Adams, Pat			▼
Au_ID	Author	Year Born	▲
1	Adams, Pat		
2	Adrian, Merv		
3	Ageloff, Roy	1943	
4	Andersen, Virginia		
5	Antonovich Michael		▼

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Lines with a lowered three-dimensional appearance

Adams, Pat			▼
Au_ID	Author	Year Born	▲
1	Adams, Pat		
2	Adrian, Merv		
3	Ageloff, Roy	1943	
4	Andersen, Virginia		
5	Antonovich Michael		▼

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Lines with a raised three-dimensional appearance

Adams, Pat			▼
Au_ID	Author	Year Born	▲
1	Adams, Pat		
2	Adrian, Merv		
3	Ageloff, Roy	1943	
4	Andersen, Virginia		
5	Antonovich Michael		▼

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Lines with a lowered three-dimensional appearance and a flat line

Adams, Pat			▼
Au_ID	Author	Year Born	▲
1	Adams, Pat		
2	Adrian, Merv		
3	Ageloff, Roy	1943	
4	Andersen, Virginia		
5	Antonovich Michael		▼

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

Lines with a raised three-dimensional appearance and a flat line

Adams, Pat			▼
Au_ID	Author	Year Born	▲
1	Adams, Pat		
2	Adrian, Merv		
3	Ageloff, Roy	1943	
4	Andersen, Virginia		
5	Antonovich Michae		▼

Use the [ListApplyTo](#) and [LineApplyTo](#) properties to specify where the lines are displayed. For more information on using the ListApplyTo and LineApplyTo properties, see [Applying Properties to Specific Parts of the Control](#) and [Applying Properties to Specific Lines](#).

[Print](#)

[Copy](#)

[Close](#)

To add lines to a list

Designer Page

1. If you want to specify where lines are added,
 - a. On the [Line subtab of the ApplyTo designer page](#), select an item from the List Apply To drop-down list box.
 - b. If you chose 9 - Single Col Header or 12 - Single Item to add lines to a specific cell, column, column header, or row, either select the column from either the Col, Col Name, or Col ID drop-down list box and type the row in the Row box as appropriate, or click the cell, column, or row in the preview area.
 - c. If you chose 10 - Single Group Header or 11 - Single Group to add lines to a specific group or group header, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
2. If you do not want the line style applied to all lines, on the Line subtab of the ApplyTo designer page, select where you want the line style applied from the Line Apply To drop-down list box.
3. Add lines by choosing the line style.
 - a. On the Line subtab of the ApplyTo designer page, select the [LineStyle](#) property from the properties list box.
 - b. Select an item other than 1 - None from the drop-down combo box under LineStyle in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To add lines to a list

Browser/Code

1. If you want to specify where lines are added, at run time,
 - a. Set the [ListApplyTo](#) property.
 - b. If you want to display the lines in a specific cell, column, column header, or row, set the [Col](#), [ColFromID](#), or [ColFromName](#) property and the [Row](#) property as appropriate.
 - c. If you want to display the lines in a specific group or group header, set the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Note If you specified where the lines are added in step 1, you should also perform steps 2 and 3 at run time.

2. Add lines by specifying where you want the line style applied with the [LineApplyTo](#) property.
3. Set the [LineStyle](#) property to a value other than 1 (None) to add lines.

Customizing Lines

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can customize the width and color of flat lines. You can specify the width and the highlight and shadow colors of three-dimensional lines.

If you have a multiple-column fpCombo or fpList control, you can also specify whether vertical or horizontal lines are displayed to fit the control, displayed only through the part of the list that contains items, or displayed through the complete list.

Use the [ListApplyTo](#) and [LineApplyTo](#) properties to specify which lines are customized. For more information on using the ListApplyTo and LineApplyTo properties, see [Applying Properties to Specific Parts of the Control](#) and [Applying Properties to Specific Lines](#).

[Print](#)

[Copy](#)

[Close](#)

To customize lines

Designer Page

1. If you want to customize flat lines,
 - a. Display flat lines in the control (the [LineStyle](#) property must be set to either 2 - Flat, 5 - Lowered w/ Line, or 6 - Raised w/ Line).
For instructions, see [Adding Lines](#).
 - b. If you want to specify where to customize the lines,
 - i. On the [Line subtab of the ApplyTo designer page](#), select an item from the List Apply To drop-down list box.
 - ii. If you chose 9 - Single Col Header or 12 - Single Item to customize flat lines in a specific cell, column, column header, or row, either select the column from either the Col, Col Name, or Col ID drop-down list box and type the row in the Row box as appropriate, or click the cell, column, or row in the preview area.
 - iii. If you chose 10 - Single Group Header or 11 - Single Group to customize flat lines in a specific group or group header, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
 - c. On the Line subtab of the ApplyTo designer page, select an item from the Line Apply To drop-down list box to specify which lines you want to customize.
 - d. If you want to change the line width,
 - i. Select the [LineWidth](#) property from the properties list box.
 - ii. Type the width in pixels in the box under LineWidth in the property value area.
 - e. If you want to change the line color,
 - i. Select the [LineColor](#) property from the properties list box.
 - ii. Choose the Color button under LineColor in the property value area to display the LineColor dialog box.
 - iii. Select a basic color or define your own custom color.
 - iv. Choose the OK button.
2. If you want to customize three-dimensional lines,
 - a. Display three-dimensional lines in the control (the LineStyle property must be set to either 3 - Lowered, 4 - Raised, 5 - Lowered w/ Line, or 6 - Raised w/ Line).
For instructions, see [Adding Lines](#).
 - b. If you want to specify where to customize the three-dimensional lines,
 - i. On the Line subtab of the ApplyTo designer page, select an item from the List Apply To drop-down list box.
 - ii. If you chose 9 - Single Col Header or 12 - Single Item to customize three-dimensional lines in a specific cell, column, column header, or row, either select the column from either the Col, Col Name, or Col ID drop-down list box and type the row in the Row box as appropriate, or click the cell, column, or row in the preview area.
 - iii. If you chose 10 - Single Group Header or 11 - Single Group to customize three-dimensional lines in a specific group or group header, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
 - c. On the Line subtab of the ApplyTo designer page, select an item from the Line Apply To drop-down list box to specify which lines you want to customize.
 - d. If you want to change the lines three-dimensional line width,
 - i. Select the [Line3DWidth](#) property from the properties list box.
 - ii. Type the width in pixels in the box under Line3DWidth in the property value area.
 - e. If you want to change the lines three-dimensional highlight or shadow color,
 - i. Select the [Line3DLight](#) or [Line3DDark](#) property from the properties list box.
 - ii. Choose the Color button under Line3DLight or Line3DDark in the property value area to display the Line3DLight or Line3DDark dialog box.
 - iii. Select a basic color or define your own custom color.
 - iv. Choose the OK button.
3. If you want to specify how vertical lines are displayed in a multiple-column control,

- a. On the [Misc subtab of the Appearance designer page](#), select the [ExtendRow](#) or [ExtendCol](#) property from the properties list box.
- b. Select the appropriate option button under ExtendCol or ExtendRow in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To customize lines

Browser/Code

1. To customize flat lines,
 - a. Display flat lines in the control (the [LineStyle](#) property must be set to either 2 (Flat), 5 (Lowered w/ Line), or 6 (Raised w/ Line)).
For instructions, see [Adding Lines](#).
 - b. If you want to specify where to customize the flat lines, at run time,
 - i. Set the [ListApplyTo](#) property.
 - ii. If you want to apply the line customization to a specific cell, column, column header, or row, set the [Col](#), [ColFromID](#), or [ColFromName](#) property and the [Row](#) property as appropriate.
 - iii. If you want to apply the line customization to a group or group header, set the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Note If you specified where the lines are added in steps 1.a and 1.b, you should also perform steps 1.c

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

1.e at run time.

- c. Specify which lines you want to customize with the [LineApplyTo](#) property.
 - d. Set the [LineWidth](#) property to determine the line width in pixels.
 - e. Set the [LineColor](#) property to change the line color.
 2. To customize three-dimensional lines,
 - a. Display three-dimensional lines in the control (the [LineStyle](#) property must be set to either 3 (Lowered), 4 (Raised), 5 (Lowered w/ Line), or 6 (Raised w/ Line)).
For instructions, see [Adding Lines](#).
 - b. If you want to specify where to customize the three-dimensional lines, at run time,
 - i. Set the [ListApplyTo](#) property.
 - ii. If you want to apply the line customization to a specific cell, column, column header, or row, set the [Col](#), [ColFromID](#), or [ColFromName](#) property and the [Row](#) property as appropriate.
 - iii. If you want to apply the line customization to a group or group header, set the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Note If you specified where the lines are added in steps 2.a and 2.b, you should also perform steps 2.c

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

2.e at run time.

- c. Specify which lines you want to customize with the [LineApplyTo](#) property.
 - d. Set the [Line3DWidth](#) property to specify the line width in pixels.
 - e. Set the [Line3DLight](#) and [Line3DDark](#) properties to change the highlight and shadow colors.
 3. Set the [ExtendRow](#) or [ExtendCol](#) property to specify how vertical lines are displayed in a multiple-column control.

Changing the Background Color

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can change the background color of different parts of a List Pro control.

Use the [ListApplyTo](#) property to specify where the background color is applied. For more information on using the ListApplyTo property, see [Applying Properties to Specific Parts of the Control](#).

[Print](#)

[Copy](#)

[Close](#)

To change the background color

Designer Page

1. If you want to specify where the background color is applied,
 - a. On the [List subtab of the ApplyTo designer page](#), select an item from the List Apply To drop-down list box.
 - b. If you chose 9 - Single Col Header or 12 - Single Item to apply the background color to a specific cell, column, column header, or row, either select the column from either the Col, Col Name, or Col ID drop-down list box and type the row in the Row box as appropriate, or click the cell, column, or row in the preview area.
 - c. If you chose 10 - Single Group Header or 11 - Single Group to apply the background color to a specific group or group header, either select the group from the Grp, Grp Name, or Grp ID drop-down list box or click the group in the preview area.
2. Specify the background color.
 - a. On the List subtab of the ApplyTo designer page, select the [BackColor](#) property from the properties list.
 - b. Choose the Color button under BackColor in the property value area to display the BackColor dialog box.
 - c. Select a basic color or define your own custom color.
 - d. Choose the OK button.

[Print](#)

[Copy](#)

[Close](#)

To change the background color

Browser/Code

1. If you want to specify where the background color is applied, at run time,
 - a. Set the [ListApplyTo](#) property.
 - b. If you want to apply the background color to a specific cell, column, column header, or row, set the [Col](#), [ColFromID](#), or [ColFromName](#) property and the Row property as appropriate.
 - c. If you want to apply the background color to a specific group or group header, set the [Grp](#), [GrpFromID](#), or [GrpFromName](#) property.

Note If you specified where the background color is applied in step 1, you should also perform step 2 at run time.
2. Specify the background color with the [BackColor](#) property.

Changing the List Gray Area Color

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

You can specify the color of the area between the cells and the control border, or the list gray area color. The list gray area appears when the list items do not cover the entire control area.

[Print](#)

[Copy](#)

[Close](#)

To change the list gray area color

Designer Page

1. On the [Color subtab of the Appearance designer page](#), select the [ListGrayAreaColor](#) property.
2. Choose the Color button under ListGrayAreaColor in the property value area to display the ListGrayAreaColor dialog box.
3. Select a basic color or define your own custom color.
4. Choose the OK button.

[Print](#)

[Copy](#)

[Close](#)

To change the list gray area color

Browser/Code

Specify the list gray area color with the [ListGrayAreaColor](#) property.

Displaying and Customizing Scroll Bars

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

By default, the horizontal and vertical scroll bars are shown only if needed. You can customize either the horizontal or vertical scroll bar, or both, to be

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

displayed at all times

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

never displayed

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

displayed at all times, but displayed as unavailable when not needed

You can also control the amount by which the list box scrolls when you click the left or right scroll arrow on the horizontal scroll bar. You can specify the amount in twips, pixels, number of characters, or the number of columns (if the [Columns](#) property is set to a value greater than 0).

If you are working in virtual mode, you might want to use a customized scroll bar. For more information, see [Using Virtual Mode](#).

[Print](#)

[Copy](#)

[Close](#)

To display and customize scroll bars

Designer Page

1. Specify when and how the scroll bars are displayed.
 - a. On the [Scroll subtab of the Appearance designer page](#), select the [ScrollBarH](#) or [ScrollBarV](#) property from the properties list box.
 - b. Select the appropriate option button under ScrollBarH or ScrollBarV in the property value area.
2. If you want to specify how far the list box scrolls when you click the left or right scroll arrows on the horizontal scroll bar,
 - a. To specify the unit of measure,
 - i. Select the [ScrollHScale](#) property from the properties list box.
 - ii. Select the appropriate list item from the drop-down combo box under ScrollHScale in the property value area.
 - b. To specify the number of units by which the list box scrolls,
 - i. Select the [ScrollHInc](#) property from the properties list box.
 - ii. Type the appropriate value in the box under ScrollHInc in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To display and customize scroll bars

Browser/Code

1. Specify when and how the scroll bars are displayed with the [ScrollBarH](#) and [ScrollBarV](#) properties.
2. If you want to specify how far the list box scrolls when you click the left or right scroll arrows on the horizontal scroll bar,
 - a. Specify the unit of measure with the [ScrollHScale](#) property.
 - b. Specify the number of units by which the list box scrolls with the [ScrollHInc](#) property.

Specifying the Number of Rows Displayed in the Drop-Down List

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

By default, eight rows are displayed in the drop-down list of a drop-down combo box or drop-down list box. You can specify any number of rows to display.

[Print](#)

[Copy](#)

[Close](#)

To specify the maximum number of rows to display

Designer Page

1. On the [Misc subtab of the Appearance designer page](#), select the [MaxDrop](#) property from the properties list box.
2. Type the number of rows in the box under MaxDrop in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To specify the maximum number of rows to display

Browser/Code

Specify the number of rows with the [MaxDrop](#) property.

Displaying Drop Shadows

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

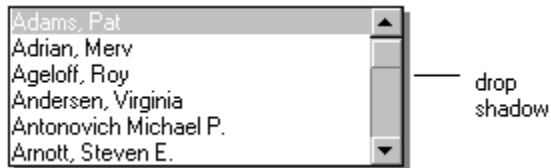
[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

List Pro controls can display drop shadows that make the controls appear raised above the window, as shown in the following figure.



Controls can display drop shadows all the time or only when they receive the focus. If the control displays a drop shadow only when it receives the focus, the area behind the control where the drop shadow appears can display a different color. The area behind the control is also visible if you set the [BorderStyle](#) property to a value other than 0 (No Border) or 1 (Single Line).

Note Remember to keep in mind the size of the drop shadow when sizing the control. The drop shadow is part of the control whether the control has the focus or not.

[Print](#)

[Copy](#)

[Close](#)

To display and customize a drop shadow

Designer Page

1. Specify when and how to display the drop shadow.
 - a. On the [Border subtab of the Appearance designer page](#), select the [BorderDropShadow](#) property from the properties list.
 - b. Select either the 1 - Always or 2 - On Focus option button under BorderDropShadow in the property value area.
2. Set the width of the drop shadow.
 - a. Select the [BorderDropShadowWidth](#) property from the properties list.
 - b. Type the width in pixels in the box under BorderDropShadowWidth in the property value area.
3. Specify the color of the drop shadow.
 - a. On the [Color subtab of the Appearance designer page](#), select the [BorderDropShadowColor](#) property from the properties list.
 - b. Choose the Color button under BorderDropShadowColor in the property value area to display the BorderDropShadowColor dialog box.
 - c. Select a basic color or define your own custom color.
 - d. Choose the OK button.
4. Specify the color of the area around and behind the control.
 - a. On the Color subtab of the Appearance designer page, select the [BorderGrayAreaColor](#) property from the properties list.
 - b. Choose the Color button under BorderGrayAreaColor in the property value area to display the BorderGrayAreaColor dialog box.
 - c. Select a basic color or define your own custom color.
 - d. Choose the OK button.

[Print](#)

[Copy](#)

[Close](#)

To display and customize a drop shadow

Browser/Code

1. Set the [BorderDropShadow](#) property to 1 (Always) or 2 (On Focus) to specify when and how to display the drop shadow.
2. Specify the width of the drop shadow in pixels by setting the [BorderDropShadowWidth](#) property.
3. Specify the color of the drop shadow by setting the [BorderDropShadowColor](#) property.
4. Specify the color of the area around and behind the control by setting the [BorderGrayAreaColor](#) property.

Customizing the Controls Borders

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Overview](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Using Predefined Appearance Styles](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Specifying the Border Appearance](#)

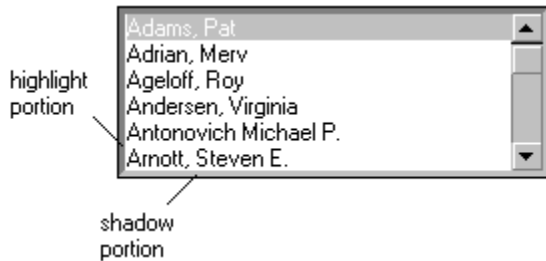
Overview

List Pro controls provide three borders that can create a three-dimensional appearance. You can set the color and width of each border to create border effects similar to most operating environment interfaces. List Pro controls provide an outline border, an outer border, and an inner border to create a three-dimensional effect. Your control can display any or all of these borders.

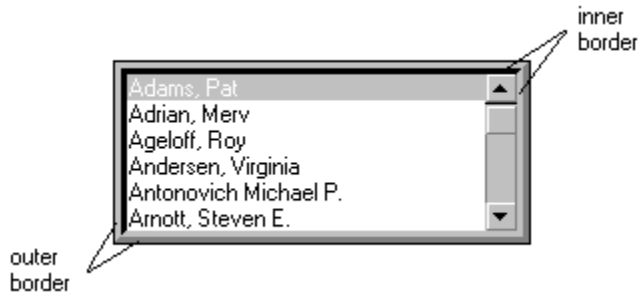
The outline border is displayed surrounding the control as shown in the following figure. You can specify the border width and color.



The outer border is displayed inside the outline border. The outer border consists of two portions, a highlight portion and a shadow portion. Using different colors for the highlight and shadow portions creates a three-dimensional effect, as shown in the following figure.



The inner border is displayed inside the outer border. Like the outer border, it consists of two portions, a highlight portion and a shadow portion. The two portions create a three-dimensional effect, as shown in the following figure.

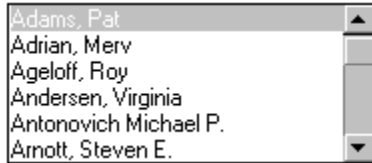


You can use some or all of these borders to create a custom appearance for your control, or you can use one of the three predefined appearance styles List Pro provides, as described in [Using Predefined Appearance Styles](#). In addition, using the inner and outer borders, you can create a frame effect for the control. For more information about creating a frame, see [Creating a Frame](#).

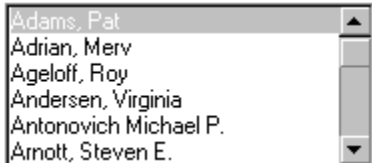
Using Predefined Appearance Styles

List Pro provides three predefined appearance styles specified by the [Appearance](#) property.

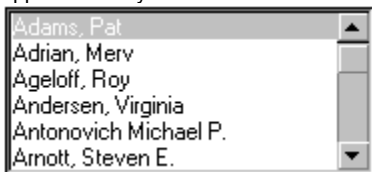
The flat appearance style provides only an outline border for the control. This appearance is similar to controls displayed in Windows 3.1 or Windows NT 3.5. An fpList control with the flat appearance style is shown in the following figure.



The three-dimensional appearance (3-D) style displays the inner and outer borders, with no outline border. The default highlight and shadow colors create the three-dimensional appearance. This appearance style is similar to controls displayed in Windows 95. An fpList control with the three-dimensional appearance style is shown in the following figure.



The three-dimensional with border appearance (3-D with Border) style displays inner, outer, and outline borders. The default highlight and shadow colors create the three-dimensional appearance. An fpList control with the three-dimensional with border appearance style is shown in the following figure.



The three predefined appearance styles are created by changing the settings for six properties. If you select one of the predefined appearance styles, the property settings for those properties are changed to create the style. The following table summarizes the properties and the settings used for each predefined style.

Caution If you have previously set any of these properties and then choose one of the predefined appearance styles, your settings will be lost.

Property	Appearance		
	Flat	3-D	3-D with Border
ThreeDInsideStyle	0 (None)	1 (Lowered)	1 (Lowered)
ThreeDInsideWidth	N/A	1	1
ThreeDOutsideStyle	0 (None)	1 (Lowered)	1 (Lowered)
ThreeDOutsideWidth	N/A	1	1
BorderStyle	1 (Single Line)	0 (No Border)	1 (Single Line)
ThreeDFrameWidth	0	0	0

If you choose a predefined appearance, you can further customize the appearance using the following instructions. However, once you change one of the six properties listed in the preceding table, the Appearance property setting changes to 0 (Custom) as the control no longer displays one of the predefined appearance styles.

Specifying the Border Appearance

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

The following instructions explain how to specify the appearance of the controls borders, either by choosing a predefined appearance style or by customizing the borders.

[Print](#)

[Copy](#)

[Close](#)

To customize the borders

Designer Page

1. If you want to use one of the predefined appearance styles,
 - a. On the [Border subtab of the Appearance designer page](#), select the [Appearance](#) property in the properties list box.
 - b. Select either the 1 - Flat, 2 - 3-D, or 3 - 3-D with Border option button under Appearance in the property value area.
If you prefer, you can further customize a predefined appearance using the following instructions.
2. If you want to display and customize the outline border,
 - a. On the Border subtab of the Appearance designer page, select the [BorderStyle](#) property from the properties list box.
 - b. Select an item other than 0 - No Border from the drop-down combo box under BorderStyle in the property value area.
 - c. To specify the border width,
 - i. Select the [BorderWidth](#) property from the properties list box.
 - ii. Type the width in pixels in the box under BorderWidth in the property value area.
 - d. To specify the border color,
 - i. On the [Color subtab of the Appearance designer page](#), select the [BorderColor](#) property from the properties list box.
 - ii. Choose the Color button under BorderColor in the property value area to display the BorderColor dialog box.
 - iii. Select a basic color or define your own custom color.
 - iv. Choose the OK button.
3. If you want to display and customize the outer border,
 - a. To display the outer border,
 - i. On the Border subtab of the Appearance designer page, select the [ThreeDOutsideStyle](#) property from the properties list box.
 - ii. Select an option button other than 0 - None under ThreeDOutsideStyle in the property value area.
 - b. To specify the width of the outer border,
 - i. Select the [ThreeDOutsideWidth](#) property from the properties list box.
 - ii. Type the width in pixels in the box under ThreeDOutsideWidth in the property value area.
 - c. To specify the color of the highlight or shadow portions of the outer border,
 - i. On the Color subtab of the Appearance designer page, select the [ThreeDOutsideHighlightColor](#) or [ThreeDOutsideShadowColor](#) property from the properties list box.
 - ii. Choose the Color button under ThreeDOutsideHighlightColor or ThreeDOutsideShadowColor in the property value area to display the ThreeDOutsideHighlightColor or ThreeDOutsideShadowColor dialog box.
 - iii. Select a basic color or define your own custom color.
 - iv. Choose the OK button.

Tip For best results, the highlight color should be a shade lighter than the background color of the form, and the shadow color should be a shade darker than the background color of the form.
4. If you want to display and customize the inner border,
 - a. To display the inner border,
 - i. On the Border subtab of the Appearance designer page, select the [ThreeDInsideStyle](#) property from the properties list box.
 - ii. Select an option button other than 0 - None under ThreeDInsideStyle in the property value area.
 - b. To specify the width of the inner border,
 - i. Select the [ThreeDInsideWidth](#) property from the properties list box.
 - ii. Type the width in pixels in the box under ThreeDInsideWidth in the property value area.
 - c. To specify the color of the highlight or shadow portion of the inner border,
 - i. On the Color subtab of the Appearance designer page, select the [ThreeDInsideHighlightColor](#) or

ThreeDInsideShadowColor property from the properties list box.

- ii. Choose the Color button under ThreeDInsideHighlightColor or ThreeDInsideShadowColor in the property value area to display the ThreeDInsideHighlightColor or ThreeDInsideShadowColor dialog box.
- iii. Select a basic color or define your own custom color.
- iv. Choose the OK button.

[Print](#)

[Copy](#)

[Close](#)

To customize the borders

Browser/Code

1. If you want to use one of the predefined appearance styles, set the [Appearance](#) property to 1 (Flat), 2 (3-D), or 3 (3-D with Border).

If you prefer, you can further customize a predefined appearance using the following instructions.

2. If you want to display and customize the outline border,
 - a. Set the [BorderStyle](#) property to a value other than 0 (No Border).
 - b. Specify the border color with the [BorderColor](#) property.
 - c. Specify the border width in pixels with the [BorderWidth](#) property.
3. If you want to display and customize the outer border,
 - a. Set the [ThreeDOutsideStyle](#) property to a value other than 0 (None).
 - b. Specify the width of the outer border in pixels by setting the [ThreeDOutsideWidth](#) property.
 - c. Specify the colors of the highlight and shadow portions of the outer border using the [ThreeDOutsideHighlightColor](#) and [ThreeDOutsideShadowColor](#) properties.

Tip For best results, the highlight color should be a shade lighter than the background color of the form, and the shadow color should be a shade darker than the background color of the form.
4. If you want to display and customize the inner border,
 - a. Set the [ThreeDInsideStyle](#) property to a value other than 0 (None).
 - b. Specify the width of the inner border in pixels by setting the [ThreeDInsideWidth](#) property.
 - c. Specify the colors of the highlight and shadow portions of the inner border by setting the [ThreeDInsideHighlightColor](#) and [ThreeDInsideShadowColor](#) properties.

Creating a Frame

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

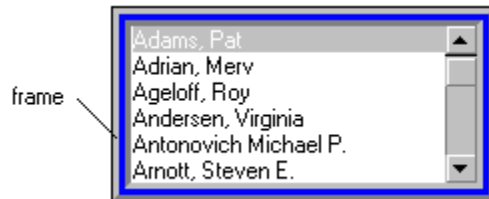
[Designer Page Instructions](#)

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

Overview

If your control displays the inner and outer borders, it can display a frame. The frame is created by displaying space between the inner and outer borders, as shown in the following figure. You can specify the size and color of the space between the borders.



For more information on displaying inner and outer borders, see [Specifying the Border Appearance](#).

[Print](#)

[Copy](#)

[Close](#)

To create a frame

Designer Page

1. Display inner and outer borders for your control.

For instructions, see [Specifying the Border Appearance](#).

2. To create the frame, specify the width of the frame as a value greater than zero.
 - a. On the [Border subtab of the Appearance designer page](#), select the [ThreeDFrameWidth](#) property in the properties list box.
 - b. Type the width in pixels in the box under ThreeDFrameWidth in the property value area.
3. To specify the frame color,
 - a. On the [Color subtab of the Appearance designer page](#), select the [ThreeDFrameColor](#) property from the properties list box.
 - b. Choose the Color button under ThreeDFrameColor in the property value area to display the ThreeDFrameColor dialog box.
 - c. Select a basic color or define your own custom color.
 - d. Choose the OK button.

[Print](#)

[Copy](#)

[Close](#)

To create a frame

Browser/Code

1. Display inner and outer borders for your control.

For instructions, see [Specifying the Border Appearance](#).

2. Set the [ThreeDFrameWidth](#) property to a value greater than zero to specify the width of the frame in pixels, thereby creating the frame.
3. Specify the color of the frame with the [ThreeDFrameColor](#) property.

Displaying Focus on the Control

PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Designer Page Instructions](#)

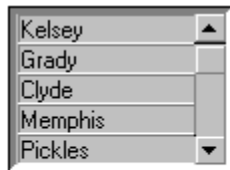
PD	RD	WR	RT	DT
✓	✓	✓	✓	✓

[Browser/Code Instructions](#)

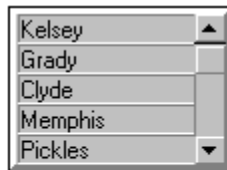
Overview

When a control has the focus, it can receive input from the user. You can show the user when a control has the focus by changing the controls three-dimensional appearance or by displaying a drop shadow. For instructions on displaying a drop shadow when the control receives the focus, see [Displaying Drop Shadows](#).

Controls that display a three-dimensional appearance can reverse the highlight and shadow colors when they receive the focus, as shown in the following figure.



Control does not have the focus



Control has the focus

Note The control must have a three-dimensional appearance to use this feature. For more information on creating a three-dimensional appearance, see [Customizing the Controls Borders](#).

[Print](#)

[Copy](#)

[Close](#)

To display focus on the control by changing its three-dimensional appearance

Designer Page

1. On the [Border subtab of the Appearance designer page](#), select the [ThreeDOnFocusInvert](#) property from the properties list.
2. Select the True option button under ThreeDOnFocusInvert in the property value area.

[Print](#)

[Copy](#)

[Close](#)

To display focus on the control by changing its three-dimensional appearance

Browser/Code

Set the [ThreeDOnFocusInvert](#) property to True to invert the colors used for three-dimensional effects.

