



ComponentOne® FlexGrid for WinForms

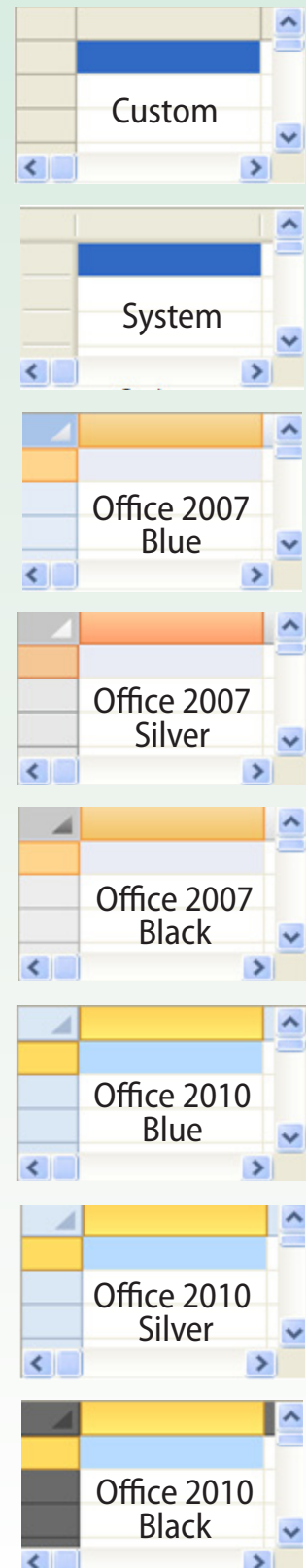
	1	Name	Product	Last Backup
		ComponentOne	Doc-To-Help	11/16/2010
2		ComponentOne	Studio for Silverl	11/2/2009
		ComponentOne	Studio for WPF	8/19/2010
		ComponentOne	Studio for ASP.N	5/18/2010
	3	ComponentOne	Studio for iPhon	1/11/2010
		ComponentOne	Studio for Active	9/22/2009
		ComponentOne	OLAP for WinFo	9/25/2009

1. Fixed RowAppearance set by **Styles.Fixed**; Count set by **Rows.Fixed****2. Fixed Column**Appearance set by **Styles.Fixed**; Count set by **Cols.Fixed****3. Frozen Column**Appearance set by **Styles.Frozen**; Count set by **Cols.Frozen/Rows.Frozen****4. Selection**Appearance set by **Styles.Highlight**; Properties used for selection: **Row, Col, RowSel, ColSel** Method used for selection: **Select(...)****5. Empty Area**Appearance set by **Styles.EmptyArea****6. Scrollbar**Set **ScrollableControl.ScrollBars** to *None, Both, Vertical, or Horizontal***7. Scrollable Area**Appearance set by **Styles.Normal**; Row/Column count set by **Rows.Count/Cols.Count**

In the illustration above, the grid has been populated by setting its **DataSource** property to a list of products. The empty area of the grid is visible because there aren't enough columns to fill it, and the data in the middle column is truncated.

After using the grid's designer to set **AutoSize** = TRUE, **ExtendLastCol** = TRUE, and **Styles.EmptyArea** to **BackColor**, the grid appearance changes to the following (columns are automatically sized to fit the content, and the last one extends to fill the control):

	Name	Product	Last Backup
	ComponentOne	Doc-To-Help	11/16/2010
	ComponentOne	Studio for Silverlight	11/2/2009
	ComponentOne	Studio for WPF	8/19/2010
	ComponentOne	Studio for ASP.NET AJAX	5/18/2010
	ComponentOne	Studio for iPhone	1/11/2010
	ComponentOne	Studio for ActiveX	9/22/2009
	ComponentOne	OLAP for WinForms	2/25/2009
	ComponentOne	IntelliSpell	10/05/2010

Visual Styles

Printing

```
// print the grid directly to the printer
_flex.PrintGrid("my grid")

// shows a print preview dialog
_flex.PrintGrid("my grid", PrintGridFlags.
ShowPreviewDialog)
```

Overloads are available to specify the document name, various printing flags, headers, and footers.

Getting/Setting Data

```
// get data from a grid cell // column
may be specified by index, // by name,
or as a reference
var xxx = _flex[row, col];

// assign data to a grid cell
// column may be specified by index,
// by name, or as a reference
_flex[row, col] = xxx;
```

Clipboard Support

```
// enable automatic clipboard support
// (cut/copy/paste)
_flex.AutoClipboard = true;

// configure clipboard copy mode
_flex.ClipboardCopyMode =
ClipboardCopyModeEnum.DataOnly;

// clipboard methods:
_flex.Cut();
_flex.Copy();
_flex.Paste();
```

Resources

Documentation: View the C1FlexGrid PDF or online help for tips, tutorials, and task-based help instructions.

Forums: Visit the C1FlexGrid forum to provide comments, ask questions, or share knowledge about C1FlexGrid with the ComponentOne community.

Samples:

- Click [Start](#) | [All Programs](#) | [ComponentOne](#) | [Studio for WinForms](#) | [Samples](#) | [FlexGrid Samples](#) to access the Sample Explorer installed with the Studio
- Download C1FlexGrid samples from the [Samples Gallery](#)

Importing/Exporting

```
// save or load grid to files or streams in various formats (CSV, TXT, XLS, etc)
_flex.SaveGrid("c:\grid.txt", FileFormatEnum.TextTab [, flags, encoding]);
_flex.LoadGrid("c:\grid.txt", FileFormatEnum.TextTab [, flags, encoding]);

// save or load grid to Excel files or streams (XLS, XLSX) with additional options
_flex.SaveExcel("c:\book1.xls" [, sheetName, flags, printerSettings]);
_flex.LoadExcel("c:\book1.xls" [, sheetName, flags]);

// save and load grid to XML files/streams preserving all data and formatting
_flex.WriteXml(file / stream / XmlWriter);
_flex.ReadXml(file / stream / XmlReader);
```

Selection/Looping

```
// get or set the selection mode (range, row, listbox, etc.)
_flex.SelectionMode = SelectionModeEnum.ListBox;

// get the current selection CellRange sel = _flex.Selection;

// select a range of cells
_flex.Select(rowStart, colStart, rowEnd, colEnd [, scrollIntoView]);

// select a row (in ListBox selection mode) _flex.Rows[row].Selected = true;

// get all selected rows RowCollection selRows = _flex.Rows.Selected;

// events fired before and after the selection changes BeforeSelChange,
AfterSelChange

// loop through all data rows:
for (int index = _flex.Rows.Fixed; index < _flex.Rows.Count; index++) {
// get the item bound to this row (often a DataRowView object) var boundItem =
_flex.Rows[index].DataSource; }

// loop through all selected rows:
foreach (Row row in _flex.Rows.Selected) { // get the item bound to this row (often
a DataRowView object) var boundItem = row.DataSource;
}
```

Grouping/Summarizing (Subtotals)

```
// create subtotal (node) rows to group and summarize data
_flex.Subtotal(AggregateEnum aggType [, level, groupFrom, groupTo, totalOn,
caption]);
```

Subtotals are not updated automatically. When the data changes, you have to re-create the groups. The example below removes any existing subtotals, then creates an outline that summarizes sales (on column 3) by Product and Region (on columns 0 and 1):

```
// select a range of cells
_flex.Select(rowStart, colStart, rowEnd, colEnd [, scrollIntoView]);

// build outline with sales totals by Product and Region
_flex.Redraw = false;
_flex.Subtotal(AggregateEnum.Clear);
_flex.Tree.Column = 0; // show outline tree on first column
_flex.Subtotal(AggregateEnum.Sum, -1, -1, 3, "Grand Total"); // grand total
_flex.Subtotal(AggregateEnum.Sum, 0, "Product", "Sales");
_flex.Subtotal(AggregateEnum.Sum, 1, "Region", "Sales");
_flex.Tree.Show(1); // expand outline to level 1
_flex.Redraw = true;
```