10Tec iGrid.NET 14.0
What's New

Contents

Change tags used in this document:

[Added]	–	new feature
[Changed]	–	change in member functionality or component behavior
[Renamed]	–	member renamed
[Improved]	–	functionality improved or UI assets became better
[Optimized]	–	performance or memory usage improvement
[Removed]	–	member or functionality removed
[Fixed]	–	fixed bug or resolved problem

Release Overview

This major update of iGrid focuses primarily on improving the control drawing infrastructure. One of the key improvements in this area is related to dark mode support. Both add-ons, AutoFilterManager and PrintManager, were also updated to support dark mode.

The most significant changes are as follows:

- Separate build of the control for .NET 10
- Support for dark mode in .NET 10 out-of-the-box
- Simplified management of the iGrid rendering style
- Smoother rendering of the UI on high-resolution displays
- More options for controlling the colors used in the UI elements
- Enhancements in the grouping and summary infrastructure

In this update, we've also standardized the terminology used in member names and removed outdated terms such as "XP Styles".

Dark Mode and .NET 10

1. [Added] With the release of .NET 10, Microsoft has added support for dark mode in Windows Forms applications. This release of iGrid.NET provides a new separate build for .NET 10 with the support for dark mode provided by WinForms in .NET 10. Now all three iGrid rendering styles (OS, 3D, and flat style) are automatically rendered in dark mode if this color mode has been set for the entire application by calling the **Application.SetColorMode()** method.
2. [Added] With the addition of dark mode support, the following new terms and corresponding enumerations have been introduced:
   - color mode — a color mode used to render the iGrid UI (light/dark)
   - theme token — a semantic identifier that represents a specific visual role in the UI
   - theme area — an area inside iGrid that uses the same rendering and color rules

   The new **iGColorMode** enumeration is used to specify color modes:

```
enum iGColorMode
{
  Light,
  Dark
}
```

   The new **iGThemeToken** enumeration is used to specify visual roles of elements (tokens):

```
enum iGThemeToken
{
  StrongGridLine,
  RegularGridLine,
  SelectionBackground,
  SelectionForeground,
  SelectionBackgroundNoFocus,
  SelectionForegroundNoFocus
}
```

   The **iGThemeArea** enumeration is used to specify iGrid areas in which theme tokens can have their own visual characteristics (such as color):

```
enum iGThemeArea
{
  Cells,
  Header,
  RowHeader,
  ScrollBars
}
```

3. [Added] iGrid implements the new **GetThemeColor()** method, which returns the color of the specified theme token in the specified theme area in the specified color mode:

```
Color GetThemeColor(iGThemeArea area, iGThemeToken token,
  iGColorMode colorMode)
```

4. [Added] The iGrid.NET assembly contains the new static class **iGSysColorResolver** with the only method **ToLightMode()**:

```
static Color ToLightMode(Color color)
```

   The **ToLightMode()** method returns the light-mode RGB equivalent for a **SystemColors** value or the input color "as is" otherwise. This functionality can be useful in dark mode to know the light-mode RGB color for the specified system color, which is currently widely used in the PrintManager add-on while printing iGrids in dark mode.

5. [Improved] When running in dark mode, the AutoFilterManager add-on uses a new set of lighter images optimized specifically for dark mode.

6. [Changed] The PrintManager add-on now prints iGrid contents in light mode, even when the application is rendered in dark mode. This preserves the traditional printed appearance of tabular data on paper.

Changes in Color Properties

1. [Changed] This release of iGrid automatically chooses the colors for selected cells and grid lines depending on the rendering style and color mode if these colors are not set in the corresponding properties. These properties are:

   - Selection: the **SelCellsBackColor**, **SelCellsForeColor**, **SelCellsBackColorNoFocus**, **SelCellsForeColorNoFocus** properties of iGrid.
   - Grid lines: the grid line style properties of the **iGrid.GridLines** object property (**Horizontal**, **HorizontalLastRow**, **Vertical**, **VerticalLastCol**, etc.), the **HGridLinesStyle, VGridLinesStyle** and **SeparatingLine** properties of the **iGrid.Header** object property, the **HGridLinesStyle** and **VGridLinesStyle** properties of the **iGrid.RowHeader** object property, the **ColsEdge** and **RowsEdge** properties of the **iGrid.FrozenArea** object property, the **SeparatingLine** property of the **iGrid.Footer** object property.

   The default values of these color properties are now **Color.Empty** instead of the explicit color values used in previous versions. If you leave these properties unchanged, iGrid automatically uses the same colors in light mode as in earlier releases.

   Similarly, in AutoFilterManager the new default value of the **SelItemBackColor**, **SelItemBackColorNoFocus**, **SelItemForeColor**, **SelItemForeColorNoFocus** sub-properties of the **FilterBoxColors** object property is **Color.Empty**. This causes the colors for the current color mode to be applied automatically, similar to the iGrid control itself.

2. [Added] The following new color properties were introduced to adjust the look of glyphs inside iGrid elements when they are disabled:

- **iGrid.CellCtrlForeColorDisabled** — for glyphs inside cell elementary controls, such as ellipsis buttons and combo buttons
- **iGrid.Header.ControlsForeColorDisabled** — for glyphs inside column header elementary controls, such as combo buttons
- **iGrid.ScrollBarSettings.ForeColorDisabled** — for glyphs inside scroll bars (standard arrow buttons, custom buttons on scroll bars, etc.)

iGrid used a hard-coded predefined color for disabled glyphs in the previous versions. Now you can adjust it for every area of iGrid with the properties listed above. The default value for all these properties is **SystemColors.GrayText**, which works well in light and dark color modes.

3. [Changed] The default value of the **iGrid.Header.SortInfoColor** property was changed from **SystemColors.ControlDarkDark** to **SystemColors.GrayText**. The sort info drawn with this color looks almost the same as earlier in light mode but becomes more legible in dark mode.

New Rendering Style Management

1. [Added][Removed] This release of iGrid introduces the new **iGrid.RenderStyle** property to specify the rendering style for the 4 theme areas (cell area, column header and row header areas, and scroll bars) and the iGrid border. In the previous versions of iGrid, you needed to use the **UseXPStyles** and **Appearance** properties for these 4 areas and the **iGrid.BorderStyle** property for the border if you wanted to change the look of iGrid to 3D or flat; now you can easily change the look of the entire grid with one setting.

   As an example, look at the typical code you would use in the previous versions of iGrid to change its look to flat:

```
iGrid1.UseXPStyles = false;
iGrid1.Header.UseXPStyles = false;
iGrid1.RowHeader.UseXPStyles = false;
iGrid1.ScrollBarSettings.UseXPStyles = false;

iGrid1.Appearance = iGControlPaintAppearance.StyleFlat;
iGrid1.Header.Appearance = iGControlPaintAppearance.StyleFlat;
iGrid1.RowHeader.Appearance = iGControlPaintAppearance.StyleFlat;
iGrid1.ScrollBarSettings.Appearance = iGControlPaintAppearance.StyleFlat;

iGrid1.BorderStyle = iGBorderStyle.Flat;
```

   Now compare it to the equivalent setting in this release of iGrid:

```
iGrid1.RenderStyle = iGRenderStyle.Flat;
```

   The **iGrid.RenderStyle** property accepts one of the values from the new **iGRenderStyle** enumeration:

```
enum iGRenderStyle
{
  System,
  Classic,
  Flat
}
```

   The **System** value means that iGrid uses the OS visual styles if they are available or switches automatically to the 3D style otherwise. The **Classic** value causes iGrid to use the 3D rendering style regardless of the availability of visual styles. The **Flat** value means that flat style is used regardless of the availability of visual styles.

These 3 values can be used as an equivalent for any combination of the values in the **UseXPStyles** and **Appearance** properties available in the previous versions of iGrid. The **UseXPStyles** and **Appearance** properties have become redundant and have been removed in this version of iGrid from the following objects: **iGrid**, **iGrid.Header**, **iGrid.RowHeader**, **iGrid.ScrollBarSettings**. To upgrade your code, replace assignments to these properties with the equivalent assignment to the new **iGrid.RenderStyle** property.

Note that you can no longer set separate rendering styles for each of the themed areas. This feature, available in previous versions, was rarely used in real-world applications, as there is no point in mixing rendering styles in a single grid. Because of this, we decided to abandon its support in favor of simplifying grid rendering management with that one new property.

2. [Added][Changed] The **iGBorderStyle** enumeration, which is the base type for the **iGrid.BorderStyle** property, has been supplemented with the new value named **Auto**. This value indicates that when rendering the iGrid border, the style specified in the **iGrid.RenderStyle** property should be used. The **Auto** value is the default value for the **iGrid.BorderStyle** property now, and this change allows you to set the rendering style for the entire grid control with one assignment to the **iGrid.RenderStyle** property.

   Note that you can set a rendering style for the border that is different from the one specified in the **iGrid.RenderStyle** property by choosing the corresponding value from the **iGBorderStyle** enumeration in the **iGrid.BorderStyle** property.

3. [Renamed] The **Standard** value of the **iGBorderStyle** enumeration was renamed to **Classic** to correspond to the unified terminology in this release of iGrid in which this term is used for the 3D rendering style.

Better High-DPI Support

1. [Improved] The red arrows indicating the possible future position of the dragged column header inside the header or group box areas are now scaled depending on the screen dpi. As a result, these elements are no longer miniature and are clearly visible on high-resolution screens.

2. [Improved] Drawing of action glyphs on custom scroll bar buttons was enhanced to better support high-resolution screens.

3. [Improved] The row header glyphs indicating row states (the current row triangle, the pencil glyph for edit mode, the I-Beam glyph for uncommitted changes, etc.) are now drawn using GDI+ graphical primitives instead of rendering 11x11 resource bitmaps in the previous versions. The new approach allows for drawing crisp glyphs on high-resolution screens, as well as displaying them in alternative colors in dark mode.

Row Header Area Enhancements

1. [Added][Improved][Changed] This release of iGrid automatically sets the width of the row header area to the optimal value upon completion of initialization in the code generated by the Windows Forms designer. The optimal width considers the screen DPI and makes sure the row header glyphs are fully visible without clipping.

   This functionality is based on the two new members of the **iGrid.RowHeader** object property: the **WidthAutoSet** property and the **GetPreferredWidth()** method. The **WidthAutoSet** property contains a Boolean value indicating whether the width of the row header area is set to the optimal value within the **ISupportInitialize.EndInit()** method called by the Windows Forms designer automatically after control initialization. The **iGrid.RowHeader.GetPreferredWidth()** method calculates and returns the optimal width value for the row header area. Actually, if **iGrid.RowHeader.WidthAutoSet** is set to True (the default value), the **iGrid.RowHeader.Width**

property is set to the optimal value returned by the **iGrid.RowHeader.GetPreferredWidth()** method in the **ISupportInitialize.EndInit()** call.

Please note the following, especially if you are upgrading code from previous versions of iGrid. The **iGrid.RowHeader.Width** property has the default value of 19 pixels, but since **iGrid.RowHeader.WidthAutoSet** is True by default, this width is not used in the general case. To preserve the behavior of the previous versions, explicitly set **iGrid.RowHeader.WidthAutoSet** to False.

2. [Added] Now you can change the colors used to draw row header glyphs with the following 3 new color properties implemented in the **iGrid.RowHeader** object property: **GlyphColor**, **GlyphErrorCircleColor**, **GlyphErrorMarkColor**. The **GlyphColor** property specifies the color of the glyphs not related to errors (current row triangle, pencil, uncommitted changes, and new row). The **GlyphErrorCircleColor** and **GlyphErrorMarkColor** properties specify the colors used to draw glyphs related to errors — the general error and current row error glyphs. They specify the colors of the filled circle and mark inside error row header glyphs. The default values for these 3 properties are **Color.Empty**, which means that the default colors are used depending on the current color mode.
3. [Improved] In the system and flat rendering styles, row headers are now drawn with a 1-pixel indent for their contents on both the left and right sides. This small change improves the visual perception of row header contents and is especially beneficial for the standard row header glyphs.
4. [Improved] The pencil glyph is not flipped horizontally in right-to-left mode and retains its traditional appearance with the tip in the bottom left corner.

Grouping and Summary Infrastructure Improvements

1. [Optimized] Grouping is performed about 10% faster if group summaries are defined. This applies to both interactive grouping and grouping from code using the **iGrid.Group()** method.
2. [Improved] In the **AfterAutoGroupRowCreated** event arguments, the **AutoGroupRowIndex** and **GroupedRowIndex** fields now return row indexes that remain valid after grouping is completed and can therefore be saved for later use. In the previous version, these fields returned temporary row indexes that could only be used to access the rows from within the event handler.
3. [Improved] The item counter in automatic group row titles is updated automatically now after adding or removing rows.
4. [Improved] The **iGrid.Group()** method no longer updates the footer summaries because in the vast majority of cases regrouping cells does not affect these summaries. If you need this, you can call the **iGrid.UpdateFooterSummaries()** method from an event handler of the **AfterContentsGrouped** event of iGrid.
5. [Added] iGrid now provides a new overloaded version of the **EndUpdate()** method:

```
void EndUpdate(bool updateSummaries)
```

This overload lets you specify whether group and footer summaries should be recalculated when updates are re-enabled by calling **EndUpdate()**. Calling **EndUpdate()** without parameters always recalculates these summaries. The new overload is useful when you have finished updating the grid and do not need summary recalculation.

6. [Improved] The **iGrid.Rows.AutoHeight()** and **iGrid.Cols.AutoWidth()** methods no longer recalculate summaries if updates are enabled.

Changes to the IiGControlPaint Interface

1. [Added] The **IiGControlPaint** interface defines the following new method iGrid can use to retrieve color values for tokens:

```
Color GetThemeColor(iGThemeArea area, iGThemeToken token,
  iGColorMode colorMode);
```

   To cause iGrid to use the colors returned by this method, return the new **iGControlPaintFunctions.ThemeColors** flag from the **SupportedFunctions** property in an object implementing **IiGControlPaint**.

   If you upgrade code for the previous versions of iGrid, simply add an empty implementation of this method to enable compilation for the new version.

2. [Improved] All **Draw*()** methods of the **IiGControlPaint** interface (such as **DrawCheckBox** or **DrawScrollBar**) were supplemented with the **grid** parameter of the **iGrid** type and the **colorMode** parameter of the **iGColorMode** type:

```
void DrawCheckBox(iGrid grid, Graphics g, int x, int y, int width, int height,
  CheckState checkState, iGControlState controlState, iGColorMode colorMode);

void DrawScrollBar(iGrid grid, Graphics g, int x, int y, int width,
  int height, iGScrollBarPart scrollPart, iGControlState controlState, bool
hoverEffect, iGColorMode colorMode);
…
```

   When iGrid calls these methods to draw its parts, it passes a reference to itself in the **grid** parameter and the color mode for rendering in the **colorMode** parameter. This allows you to adjust the drawing if required (for example, you can retrieve property values of the grid control in which custom drawing is performed).

   To upgrade your code from the previous versions of iGrid, add the **grid** parameter of the **iGrid** type as the first parameter and the **colorMode** parameter of the **iGColorMode** type as the last parameter to all existing **Draw*()** methods in a class implementing the **IiGControlPaint** interface.

3. [Added][Removed] The **OffsetScrollBarCustomButtonImageWhenPressed** property of the **Boolean** type was replaced with the new **PressedButtonContentOffset** property of the **Point** type. When implemented in a class, it returns a **Point** value whose **X** and **Y** properties specify the horizontal and vertical offset, in pixels, to apply to the content of a pressed button-like element.

   This **PressedButtonContentOffset** property is now used both for scroll bar custom buttons and ellipsis buttons, providing the same content offset effect when the corresponding UI element is pressed. If you used this effect in your custom control paint objects, replace the old property implementation with a new one that returns a Point(1, 1) instance for backward compatibility.

4. [Added] The **DrawHeader()** method has the new Boolean parameter **hotBackgroundEnabled**. When iGrid calls this method, it passes True in the **hotBackgroundEnabled** parameter if the **iGHdrHotTrackFlags.Background** flag is specified in the **HotTrackFlagsVisualStyle, HotTrackFlagsClassicStyle**, or **HotTrackFlagsFlatStyle** property of the **iGrid.Header** object property.

5. [Removed] The obsolete **ControlsBackColor**, **ControlsForeColor**, and **ControlsDisabledForeColor** properties were removed from the **IiGControlPaint** interface. The corresponding **ControlsColors** flag was also removed from the **iGControlPaintFunctions** enumeration. To upgrade code from the previous versions, simply remove these members from your code. It is safe to do because they are no longer used by iGrid.

Other Changes and Enhancements

1. [Improved][Renamed] The **DrawActionIcon()** method from the previous versions of iGrid was superseded with the new **DrawActionGlyph()** method, providing the following enhancements:

   - The method name including the term "glyph" better indicates the nature of the method — it draws pictorial representations for actions using GDI+ code but not predefined icon resource. This corresponds to the iGrid terminology in which the term "glyph" is used for the same purpose (for example, like in the **DrawRowHdrGlyph()** method).
   - The **DrawActionGlyph()** method can draw the glyphs for actions that were not drawn in the previous releases with the **DrawActionIcon()** method: **GoPrevCell**, **GoNextCell**, **GoPrevRow**, **GoNextRow**, **GoFirstCell**, **GoLastCell**. This also allows you to display the corresponding glyphs in custom scroll bar buttons with the corresponding associated actions.
   - The **DrawActionGlyph()** method allows you to specify the color of the glyph.
   - The drawing code of action glyphs was improved to provide better legibility on displays with various pixel density, including 4K displays.

   The signature of the **DrawActionGlyph()** method was changed compared to the **DrawActionIcon()** method to provide the ability to specify the glyph color:

   ```
   void DrawActionIcon(Graphics g, Rectangle bounds, iGActions action,
     bool enabled)

   void DrawActionGlyph(Graphics g, Rectangle bounds, iGActions action,
     Color color)
   ```

   The **DrawActionIcon()** method from the previous versions was mainly used by the internal infrastructure of iGrid to draw glyph actions on custom scroll bar buttons, so that most likely it was not used in your code and this method replacement change will not affect your code. If you used **DrawActionIcon()** for some reason in code for the previous versions of iGrid, you can replace it with calls to the new **DrawActionGlyph()** method when switching to this release of iGrid. To provide backward compatibility, you can use the color stored in the **iGrid.ScrollBarSettings.ForeColor** property when drawing enabled action glyphs and the color stored in the **iGrid.ScrollBarSettings.ForeColorDisabled** property when drawing disabled action glyphs.

2. [Improved][Changed] The implementation of the **iGPenStyle** class, which is used as the type of all properties specifying look of various grid lines, was changed. Its **Color**, **Width**, and **DashStyle** properties, which were previously class fields, are now implemented as true properties. This was necessary for the improved internal iGrid infrastructure and to control the value of the **Width** property in all possible use scenarios. Now, if you try to assign a negative value to this property in design time or run time, an **ArgumentOutOfRangeException** exception will be generated immediately with a corresponding message.

   At design time, the **iGPenStyle.Color** property now displays the unset value (**Color.Empty**) in the property grid as "NotSet" like all other color properties of iGrid (an empty string was shown in the previous versions of iGrid).

   The existing code does not require any changes when upgrading to this release of iGrid, unless you have used .NET reflection to work with the fields/properties of the **iGPenStyle** class.

3. [Improved][Renamed] The public **iGManagerCompare** class, which provided the only static method **CompareObjects()** implementing the cell value comparison logic used by iGrid, was replaced with an equivalent implementation in the new public **iGValueComparer** class. This class also implements the standard .NET **IComparer** and **IComparer<object>** interfaces and thus can be used in all non-generic and generic .NET collections supporting custom comparers. The class follows the singleton

object design pattern and provides its only instance with the static **Default** property. This allows you to use iGrid's value comparer in your own lists like this:

```
myValueList.Sort(iGValueComparer.Default);
```

If you called the static **iGManagerCompare.CompareObjects()** method in your code, simply replace these calls with the equivalent **iGValueComparer.Compare()** calls.

4. [Improved][Changed] The default value of the **iGrid.GroupBox.ColHdrSpace** property was changed from 6 to 7. This change provides a better overall look for the group box and especially the small red arrows indicating the possible future position of the dragged column header inside the group box. As a result, the default height of the group box area increased by 2 pixels.

5. [Improved] The shaded shaft of flat scroll bars no longer blinks when the user clicks this area. It was an effect that came from the outdated 3D scroll bars, and it was eliminated in this release to better correspond modern UI experience.

6. [Improved] An **InvalidOperationException** is now thrown instead of **ArgumentException** when attempting to remove the row text column.

7. [Improved] If the values of the **iGrid.ForeColorDisabled** and **iGPenStyle.Color** properties are unset (actually set to **Color.Empty**), they are displayed as "NotSet" in the property grid like other unset color properties of iGrid.

8. [Changed] The **DrawFooterRowHdrContents()** method now implements the ability to draw the background of the footer row header element and to turn this ability on/off with the new Boolean **drawBackground** parameter.

9. [Renamed] The properties of the **iGrid.Header** object whose names begin with "HotTrackFlags" have been renamed to match the unified terminology of rendering style names in this release. The **HotTrackFlagsXPStyles** property is now called **HotTrackFlagsVisualStyle**, **HotTrackFlagsStyle3D** has been renamed to **HotTrackFlagsClassicStyle**, and **HotTrackFlagsStyleFlat** has been renamed to **HotTrackFlagsFlatStyle**.

10. [Renamed] In the previous versions iGrid provided the following events related to selection change: **SelectionStartChange**, **SelectionChanging**, **SelectionEndChange**, **SelectionChanged**. The problem with these names was that they referred to different processes and could be very confusing for a developer. For example, a developer might think that the **SelectionChanging** and **SelectionChanged** events refer to a change in the selection status of a cell. In fact, the **SelectionStartChange**, **SelectionChanging**, and **SelectionEndChange** events pertain to the interactive drag-select operation, whereas the **SelectionChanged** event notifies of any change in cell selection, including those made programmatically.

    To avoid confusion, events related to a drag select operation have been renamed in this release of iGrid. The new names for the **SelectionStartChange**, **SelectionChanging**, and **SelectionEndChange** events are **DragSelectStart**, **DragSelectRangeChange**, and **DragSelectEnd** respectively. The event arguments classes for these events, **iGSelectionStartEndChangeEventArgs** and **iGSelectionChangingEventArgs**, were renamed to **iGDragSelectStartEndEventArgs** and **iGDragSelectRangeChangeEventArgs** respectively. To upgrade your existing code to work with the new release of iGrid, simply rename the relevant elements in your code.

11. [Improved] The design-time row and column collection editors have been improved. In the column collection editor, the column key is now shown before the header text, consistent with the representation used for rows. Also, line breaks in multi-line column headers and cell texts are now replaced with spaces so the full text is displayed on a single line.

Bug Fixes

1. [Fixed] iGrid did not update cell and row selection correctly after changing the values of the **RowMode**, **SelectionMode**, and **RowSelectionInCellMode** properties.
2. [Fixed] The **AutoWidth()** method for a column did not adjust the column width correctly if the column showed tree and tree buttons were visible or hidden (not absent).
3. [Fixed] If a group row was the current cell in row mode, it was highlighted with the colors specified in the **CurCellBackColor** and **CurCellForeColor** properties of iGrid regardless of the value of the **RowModeHasCurCell** property. Now it happens only if **iGrid.RowModeHasCurCell** equals True.
4. [Fixed] An exception occurred when attempting to remove the first row by its key.
5. [Fixed] Combo buttons in column headers were drawn as full buttons with the button background in disabled iGrid.
6. [Fixed] The content of scroll bar custom buttons and ellipsis buttons in the 3D rendering style was not shifted when pressed like other button elements (combo buttons, scroll bar arrow buttons, etc.)
7. [Fixed] If the row header area was rendered with the 3D style, the pressed row header was not drawn with the pressed effect.
8. [Fixed] The hot-track flags specified in the **HotTrackFlagsVisualStyle**, **HotTrackFlagsClassicStyle**, and **HotTrackFlagsFlatStyle** properties of **iGrid.Header** (named **HotTrackFlagsXPStyles**, **HotTrackFlagsStyle3D**, and **HotTrackFlagsStyleFlat** in the previous versions) did not work correctly.
9. [Fixed] A memory leak issue when drawing the flat iGrid border was fixed.
10. [Fixed] If the rendering style of iGrid was changed to 3D or flat, drop-down lists in cells continued to use visual styles for its scroll bars.
11. [Fixed] The **CustomGroupTitle** event could be raised twice for the same automatic group row during grouping if group summaries were defined.
12. [Fixed] The **iGrid.RowHeader.AutoWidth()** method considered the unneeded thickness of the vertical grid line in the row header when it was rendered using the 3D style. The same problem for the row header horizontal grid line was fixed in iGrid methods calculating the optimal row height (**iGrid.GetPreferredRowHeight()**, **iGrid.Rows.AutoHeight()** and the like).
13. [Fixed] Cells in invisible rows or columns could not be selected programmatically by setting the **iGCell.Selected** property to True. Invisible rows also could not be selected programmatically by setting the **iGRow.Selected** property to True.