# 10Tec iGrid ActiveX 4.x
# What's New in the Latest Builds

Keywords used to classify changes:

- [New] – a totally new feature;
- [Change] – a change in a member functionality or interactive behavior;
- [Fixed] – a fixed bug or solved problem;
- [Removed] – a member was completely removed;
- [Enhancement] – some functionality was enhanced;
- [Optimization] – a feature has speed improvements;
- [Renaming] – a member was renamed.

## v4.70, build 0153 | 2011-May-19

1. [Fixed] iGrid might crash if you double-clicked the area free of cells.
2. [Fixed] If you grouped the grid by two or more columns, the contents of group rows might be the same for several adjacent group rows on different hierarchy levels.
3. [Fixed] iGrid might generate an error if you removed rows in its **MouseDown** event.

## v4.70, build 0150 | 2011-Apr-15

4. [New] A new Boolean cell property, **CellSelectable**, was implemented. Using this property, you can prevent individual cells from being selected and accessible with the mouse or keyboard. To do that, just change its default value True to False. The **CellObject** also was supplemented with the corresponding property **bSelectable**.
5. [New] To control whether the current cell selection will "jump" over non-selectable cells, a new Boolean property **SkipNonSelectableCells** was implemented. It's default value is True. This means that the current cell selection will be moved to the next available cell skipping all non-selectable cells on the normal way of moving. If you set this property to False, the current cell remains the same if the next cell is non-selectable.
6. [Removed] The **bIncludeInSelect** option used with the **AddCol** method was removed as now the same effect can be achieved with the **CellSelectable** property. The only exclusion is that in row mode you can no longer select rows by clicking the cells in the columns excluded from selection, but it can be easily emulated in the **MouseDown** event if you need this functionality. The start column for row text cells depends now on one property, **RowTextStartCol**, but not also from first columns excluded from selection using the **bIncludeInSelect** option.
7. [New] A new **CellDynamicIcons** event was implemented:

```
Public Event CellDynamicIcons( _
   ByVal lRow As Long, ByVal lCol As Long, ByVal vValue As Variant, _
   ByRef btImageList As Byte, ByRef iIcon As Integer, _
   ByRef btExtraImageList As Byte, ByRef iExtraIcon As Integer)
```

   It is used the same way as the **CellDynamicText** event, but allows you to set both cell icons and/or their source image lists on the fly.

8. [New] Now iGrid supports the DEL and BACKSPACE keys which became a de facto standard in many grid applications. They are used to clear cell contents.

DEL erases the current cell contents without putting iGrid into edit mode. BACKSPACE – in contrast to DEL – starts editing, then also erases the cell contents so you see the blinking caret and can enter new text. In the case of BACKSPACE you can still revert to the original cell contents until you commit editing (say, by pressing the ESC key) – what is not possible in the case of DEL when editing isn't started at all.

The other difference between DEL and BACKSPACE in this case is that DEL clears all selected cells while BACKSPACE puts only the current cell into edit mode.

Note that pressing the DEL key also raises the **RequestEdit** event, and if a cell is protected from editing in this event, its contents won't be cleared by DEL.

9. [Enhancement] The previous builds of iGrid guaranteed that group rows created during automatic grouping were created in a loop from bottom to top rows of iGrid, but the order of creation of several group rows for next group values (when we needed two or more adjacent group rows) was not defined. Now all adjacent automatic group rows and the corresponding **AfterAutoGroupRowCreated** events are also issued from bottom to top.

10. [Enhancement][Fixed] In the previous builds of iGrid 4.x when you started the text editing in a cell and the height of the cell was not enough to display the text without clipping, the text in the cell editor might disappear at all. This could happen, for instance, after you had increased the grid or cell font, but didn't correct the row heights correspondingly.

    Starting from this build, iGrid uses another behavior. In such cases iGrid increases the height of the cell text editor to the value which allows one text row be fully visible. Sure the cell text editor can cover a row (rows) below depending on the font size, but this behavior is much better for the user as they can see at least one full row of text and edit the cell without any problems.

    This is an enhancement compared to iGrid 1.x-3.x, when we saw the cell text clipped by the next row during editing. This behavior is a kind of standard functionality used by such widely used apps as MS Excel.

11. [Enhancement][Fixed] In the previous builds the edited text was shifted by several pixels to the right against the real cell text, and there was also the same unused right margin in the cell text editor. Now the edited text appears exactly over the cell text.

12. [Enhancement] The previous builds of iGrid didn't guarantee that the **AfterSelectionChange** event related to cell selection status change was raised after the information about the current cell (the **CurRow**/**CurCol** properties) had been updated. In this build iGrid always reports the actual new values in the **CurRow** and **CurCol** properties if you access them in the **AfterSelectionChange** event.

13. [Change] When you press CTRL+X or issue an interactive command to cut the contents of the selected cells, the existing selection isn't cleared (the behavior similar to MS Excel).

14. [Renaming] The **iIconIndex** and **iExtraIconIndex** of the **CellObject** were renamed to **iIcon** and **iExtraIcon** respectively to conform with the general name system (the other cell icon related properties don't use the "Index" word).

15. [Fixed] Now the **CurCellChange** event is raised in every case when the values returned by iGrid's **CurRow** and/or **CurCol** properties are changed. This happens even if the current cell isn't changed explicitly – for instance, if you insert a new row before the row with the current cell so the **CurCol** property will return a new value.

16. [Fixed] A series of problems with raising selection related events (such as **BeforeSelectionChange** and **AfterSelectionChange**) was fixed. This concerns both iGrid members called from code (such as the **MoveRow** method or **CellSelected** property) and interactive actions performed by the user with the mouse and/or keyboard.

17. [Fixed] When you removed some last columns by decreasing the **ColCount** property, and the current cell was in the removed columns, the **CurCellChange** event wasn't raised and the **CurCol** property wasn't set to 0.
18. [Fixed] A serious bug when iGrid might crash after subsequent calls to **ColCount** when last columns were removed was fixed.
19. [Fixed] iGrid might crash when you assigned a new value to the **CurRow** property after the grid had been repopulated from a recordset with a less number of rows using the **FillFromRS** method.
20. [Fixed] iGrid might fail if you removed the last row in the **DblClick** event.
21. [Fixed] iGrid might draw selection from the left edge of the grid covering the level indent area for rows with normal cells with non-zero row level values (this level indent area should not be included into selection).
22. [Fixed] The **RequestEdit** event wasn't raised for the CTRL+X keyboard combination when the selected cells were put into the clipboard (in fact, when their contents change or editing happened).

## v4.60, build 0135 | 2010-Dec-24

23. [New] The main new feature in this release of iGrid is the ability to draw tree lines for the built-in trees. You can enable them using a new **TreeLines** property which accepts one of the values of a new **ETreeLines** enumeration:

```
Enum ETreeLines
   igTreeLinesNone = 0
   igTreeLinesDotted = 1
   igTreeLinesSolid = 2
End Enum
```

    By default, iGrid does not display tree lines (**igTreeLinesNone** is the default value for **TreeLines**), but if you set this property to another value, you will see them. As you can see, two styles are available – the standard dotted line (**igTreeLinesDotted**) or solid (**igTreeLinesSolid**).

24. [New] The color of the tree lines can be controlled through a new **TreeLinesColor** property of the OLE_COLOR data type. The default value for this property is **vb3DShadow**.
25. [New] This release of iGrid allows you to use your own tree button images instead of the standard ones.

    You can redefine the tree button images for individual or all rows. It is done with a new **CustomDrawTreeButtonRequest** event:

```
Event CustomDrawTreeButtonRequest(ByVal lRow As Long, _
   ByRef bCustomDraw As Boolean, ByRef lCustomSize As Long)
```

    If a row requires a custom tree button, check the **lRow** parameter of this event and set the **bCustomDraw** parameter to True. If you need to draw your own tree buttons for all rows, simply set **bCustomDraw** to True in this event without analyzing **lRow**.

    If you set **bCustomDraw** to True, also specify the size of your custom button with the **lCustomSize** parameter (iGrid works only with square tree buttons). Note that each row can have a custom tree button of its own size. We also recommend that you use tree buttons with odd sizes if you wish to have good look when you show tree lines in the grid – then the tree lines will be centered vertically directly at the middle line of the tree buttons.

    Once you specified custom tree button drawing in a row, iGrid will generate the following new event to draw your own custom tree button:

```
Event CustomDrawTreeButton(ByVal lRow As Long, ByVal hdc As Long, _
   ByVal lLeft As Long, ByVal lTop As Long, _
   ByVal lRight As Long, ByVal lBottom As Long, _
   ByVal bExpanded As Boolean)
```

You draw your own image on the device with the **hdc** handle using the standard Win32 API functions. The **bExpanded** parameter will help you to know the state of the drawn tree button.

26. [New] A new Boolean read-only property **RowHasChildren** was implemented. It has the following syntax

```
Property Get RowHasChildren(ByVal vRow As Variant) As Boolean
```

and indicates whether the specified row has child rows (applicable for both group rows and tree nodes).

27. [New] The **Combos** collection property has a new **Clear** method you can use to remove all defined combo lists at one go.
28. [Enhancement] The tree button including the corresponding tree lines segments isn't included into selection for the cells in the tree column.
29. [New] The **ScrollBarObject** has a new Boolean property called **MouseWheelScrollsWhenHidden**. In iGrid, when you hide a scroll bar (by setting its **DisplayMode** property to **igScrollBarDisplayNever**), the grid cannot be scrolled using the mouse wheel. This is the default behavior which corresponds to the default value of False of the **MouseWheelScrollsWhenHidden** property. If you set this property to True, you get the ability to scroll the grid in the corresponding direction using the mouse wheel even if the scroll bar is hidden.
30. [Fixed] If a scroll bar was disabled, you could still scroll the grid in the corresponding direction for some distance using the mouse wheel.
31. [Fixed] If the grid had been scrolled horizontally and you right-clicked a column header to display the built-in header context menu, it was displayed at the far right but not near the mouse pointer.

## v4.50, build 0124 | 2010-Nov-05

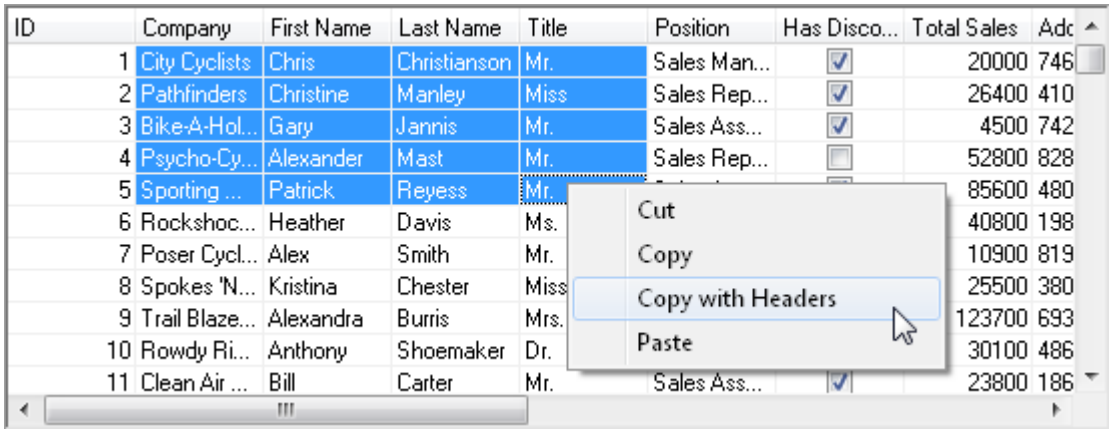
1. [Fixed] The **SelItems** object property might return an improper list of selected items in row mode.

## v4.50, build 0123 | 2010-Sep-21

1. [New] The **FillFromRS** method now supports the new DAO Recordset objects from the updated DAO library supplied with MS Office 2007/2010 ("Microsoft Office 12.0 Access database engine Object Library", ACEDAO.DLL).
2. [Fixed] When the classic "visual" style was used in Windows, iGrid had a serious drawing problem with check boxes, combo box buttons and header.
3. [Fixed] The **SelectAll** method and SHIFT+mouse click in multiselection mode didn't select row text cells.

## v4.50, build 0120 | 2010-Jul-31

1. [New] This build of iGrid introduces built-in copy/paste functionality enabled by default. New Cut, Copy and Paste commands can be accessed through the built-in context menu displayed by the right mouse click, or corresponding keyboard shortcuts:




These copy/paste commands allow you to cut or copy the contents of the selected cells into the Windows Clipboard and paste them into other Windows applications. If you copy (or cut) and paste cells within the same iGrid control, the cells are copied with all formatting options you set (colors, alignment, etc) and icons if they were defined. If you paste the copied iGrid cells into another iGrid or Windows application, only the cell values in the form of the Windows default string representation are transferred. The string representation of the copied data placed into the clipboard is built according to the de-facto standard used by MS Excel, which maximizes the compatibility with the MS Office apps and allows you to easily transfer your data to such applications as MS Excel for further analysis.

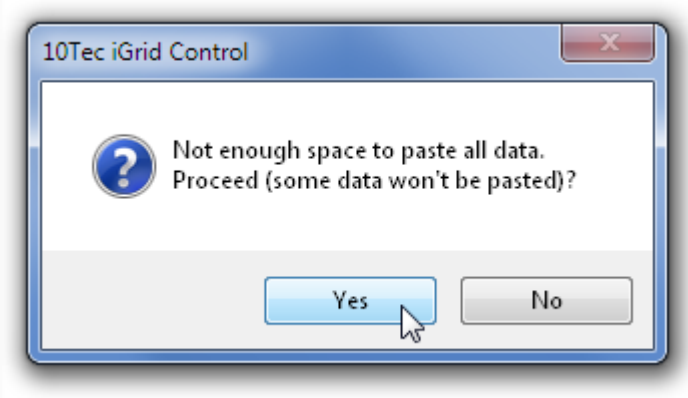
The copy/paste commands are: Cut, Copy, Copy with Headers, Paste. You see them in this order in the built-in context menu. The keyboard shortcuts for these commands are CTRL+X, CTRL+C, CTRL+SHIFT+C, and CTRL+V respectively.

The 'Copy with Headers' command also copies the texts of the column headers of the columns with the selected cells as the first data row. This feature is useful if you wish to copy into another application the column headers for your data.

The Cut and Copy commands support non-rectangular selections, and only the selected cells are copied as a "compact" rectangular range as if there were no non-selected cells between the selected ones. This is very useful in many cases, for example: you select several non-adjacent rows in row mode using CTRL+click (for instance, for further analysis in MS Excel), and then copy and paste them into MS Excel as if they were adjacent rows.

All the copy/paste commands ignore the hidden columns and rows – the user work only with the visible rows/columns as if there were no hidden cells. For instance, if the second column is hidden, and you paste a row with two cells starting from a cell in the first column, your two cells in the clipboard are pasted into the first and third columns of iGrid. Note that a hidden column is not the same as the column with zero width: a column is really hidden if its **ColVisible** property is set to False – in contrast to a column with the width of 0 (**ColWidth**) which can be made so by the user when they resize the column interactively. The same also concerns rows, but there is one more factor for them – a row can be collapsed as a row nested under a tree node or a group row, but such a row isn't considered as hidden. The row is hidden only when its **RowVisible** property is False.

If you try to paste data, and the available rows and/or columns are not enough to accommodate all the pasted data, the following alert with a corresponding question is displayed by iGrid:




In this mini-dialog, you can abandon the paste operation at all, or agree to paste as much data as iGrid can accommodate.

The Cut and Paste commands become disabled if the grid is read-only (the **Editable** property is set to False). All the copy/paste commands are unavailable if the grid is disabled (**Enabled** is False).

The captions of the Cut/Copy/Copy with Headers/Paste items of the context menu can be localized using the **UIStrings** property. Use the 12th, 13th, 14th, 15th items respectively for this purpose.

If you need to issue iGrid's copy/paste commands from code (for instance, from your own context menu), use the **DoKeyboardCommand** statement with the values which correspond the key combination for the required command. For instance, for the Copy command it will be

```
iGrid1.DoKeyboardCommand vbKeyC, vbCtrlMask
```

2. [New] The built-in copy/paste functionality can be optionally disabled using a new Boolean property named **CopyPasteEnabled**. Its default value is True, but if you set it to False, the copy/paste menu isn't displayed and all the copy/paste related keyboard combinations do not work. However, if you wish, you can still prohibit only the built-in copy/paste menu or some or all copy/paste keyboard commands using the approaches described below.

   The copy/paste built-in menu is displayed when you release the right button of the mouse. If you wish to suppress this default menu, set the **bDoDefault** parameter of the **MouseUp** event to False for the right mouse button:

```
Private Sub iGrid1_MouseUp(Button As Integer, ..., bDoDefault As Boolean)
   If Button = vbRightButton Then
      bDoDefault = False
   End If
End Sub
```

   The same approach is used to prohibit copy/paste with the keyboard, but in this case we use the **KeyDown** event. For instance, CTRL+C (copy) is suppressed with this code:

```
Private Sub iGrid1_KeyDown(KeyCode As Integer, Shift As Integer, bDoDefault
As Boolean)
   If (KeyCode = vbKeyC) And (Shift = vbCtrlMask) Then
      bDoDefault = False
   End If
End Sub
```

3. [New] The Cut and Paste commands in fact change the contents of cells, so the events **RequestEdit** and **AfterCommitEdit** are triggered when the user is copying or pasting data. The **RequestEdit** event is raised before iGrid pastes new data into each cell, and if you set its **bCancel** parameter to False, you can prohibit individual cells from pasting new data. Just note that the first 3 parameters of this event - **lRow**,

**lCol**, and **bCancel** - are effective in this case; the rest parameters (**sText**, **lMaxLength** and **eTextEditOpt**) aren't used as interactive text editing isn't used in this case.

The Cut and Paste commands change the selection status of the cells (the cut cells become unselected, the paste cells become selected automatically), so the **CellSelectionChange** event is raised for the corresponding cells. The general **BeforeSelectionChange** and **AfterSelectionChange** events are also raised as the standard wrapper events for a series of the **CellSelectionChange** events in this case.

4. [New] Now iGrid supports drag select ("drag select" is a term used to describe the method of selecting a range of cells by dragging the mouse over a block of cells). Drag select can be done while you are holding down the left or right mouse button. In the latter case the default context menu with the copy/paste commands is displayed automatically just after you have released the right mouse button to finished cell selection process, which is very convenient if you wish to select a range of cells and copy them using the minimal set of actions.

   This mode can be turned on/off using a new Boolean **DragSelect** property. It is enabled by default (i.e. the default value of this property is True).

   Note that if iGrid contains more cells than it can display in the viewport, its contents is scrolled automatically and the new cells are selected if you continue to hold down the mouse button and move the pointer out of the grid contents area.

5. [New] The iGrid cells can be selected by the right mouse click. It occurs when you press the right mouse button. The iGrid control is also activated now by the right mouse click if it was not focused.

6. [Change] The cells in the hidden rows and columns aren't selected now when the user selects a range of cells with the help of the mouse or keyboard holding down SHIFT. As one of the results of this improvement, these cells are not copied into the clipboard by the Copy command, and this is the behavior expected by the end user.

   This principle also concerns the **SelectAll** statement.

7. [New] A new **SearchString** property was implemented. It returns/sets the string typed by the user doing incremental search.

   Note that iGrid performs incremental search when you assign a new string to this property as if the user typed this string doing incremental search. In this case the new assigned string becomes the value of this property only if a match has been found; otherwise the value of the property remains unchanged.

8. [New] The **FindSearchMatchRow** method was supplemented by 2 new parameters which can be used to enhance its functionality. These are **eMatchMode** and **bMatchCase**, and the syntax of the method now looks like the following:

```
Function FindSearchMatchRow( _
   ByVal vSearchCol As Variant, _
   ByVal sSearchString As String, _
   Optional ByVal lStartRow As Long = -1, _
   Optional ByVal bVisibleRowsOnly As Boolean = True, _
   Optional ByVal bLoop As Boolean = False, _
   Optional ByVal eMatchMode As ESearchMatchMode = igSearchMatchStartsWith, _
   Optional ByVal bMatchCase As Boolean = False _
) As Long
```

   The **eMatchMode** parameter is used to specify the compare method to find the specified string. It accepts one of the following parameters of a new **ESearchMatchMode** enumeration:

```
Public Enum ESearchMatchMode
   igSearchMatchStartsWith = 0
   igSearchMatchContains = 1
   igSearchMatchEquals = 2
End Enum
```

The **bMatchCase** parameter is used to specify whether case-sensitive search is performed.

9. [New] A new Boolean property named **BrowseAllNonFrozenCols** was implemented. iGrid uses the value of this property to determine whether to scroll its contents automatically in the horizontal direction to make all the non-frozen columns available for the LEFT ARROW and RIGHT ARROW keys navigation if some of these columns were hidden "under" the frozen area (i.e. the grid had been scrolled horizontally). The default value of this property is False, which is useful if you scrolled the grid horizontally and would like to navigate through your cells using the arrow keys as if there were no columns between the frozen and non-frozen ones so the position of the horizontal scroll box remains the same when you jump from frozen columns to non-frozen and vice versa.
10. [Optimization] The **SetCurCell** method works much faster in multiselection mode. Now it is practically instant, but in the previous builds you can see a noticeable delay when it is executed on huge grids and/or on slow computers.
11. [Removed][Fixed] This version of the control no longer uses the non-immediate column resizing mode (when a special divider line is drawn throughout the grid until the moment when the user releases the mouse button while resizing a column). Now iGrid resizes the column and redraws its contents as the user is resizing the column which is much more user-friendly behavior as the users can see the result of their actions immediately. As a consequence, the **ImmediateColumnResizing** property and the **igSupFeatImmedColResizing** item of the **ESupportFeatures** enumeration were abolished.

    The previous non-immediate column resizing mode also had some problems with speed and drawing in such newest systems as Windows Vista and Windows 7, but the new immediate column resizing mode is free from these drawbacks.
12. [Fixed] A serious problem with the drawing of scroll bars in Windows Vista and Windows 7 was fixed. In these systems, if the control did not use visual styles, some parts of the scroll bars might be redrawn using visual styles chaotically when the user used mouse to control them.
13. [Fixed] Serious visual glitches of the drawing of the iGrid scroll bars in MS Access were fixed.
14. [Fixed] The **BeforeSelectionChange**/**AfterSelectionChange** events weren't raised when you changed the current cell with the **SetCurCell** method in multiselection mode.
15. [Fixed] iGrid might make improper selection when you selected rectangular ranges of cells using holding down the SHIFT key in some particular situations (for example, after scrolling the grid with the PAGE DOWN key or setting the **CellSelected** property in your code).
16. [Fixed] iGrid might crash your app if you changed the set of rows in the **MouseUp** event.
17. [Fixed] The **BeforeSelectionChange** and **AfterSelectionChange** events were not raised after the user selected/deselected cells or rows using CTRL+click in multiselection mode.
18. [Fixed] iGrid might draw its contents improperly after you moved a column using the **ColPos** method in your code
19. [Fixed] iGrid might draw its contents improperly if you had frozen columns and enlarged one of them enough fast interactively by dragging the column divider.
20. [Fixed] The iGrid contents was not scrolled horizontally when you had frozen columns and issued the **EnsureVisibleCell** method or pressed the RIGHT ARROW key.

## v4.00, build 0098 | 2010-Mar-25

1. [New] The **BeforeContentsGrouped** and **AfterContentsGrouped** events are also raised before and after the grid is ungrouped (interactively or in code using the **Ungroup** method).

2. [New] Now iGrid places the group values into the cells of the group rows created during automatic grouping so you can easily retrieve them. The values are placed in the columns corresponding to the columns a group row was created for. For instance, if you group iGrid by the cells in the second column, the cells in the second column in your group rows will contain the group values.

   Note that if you group the grid by several columns, your top-level group rows will contain only the group values from the first group column, their child group rows will contain the values only from the 1st and 2nd group columns, etc. The following code snippet demonstrates this concept and gives you an idea of how you can retrieve the group values for group rows on different levels when you click them:

```
Private Sub iGrid1_Click(ByVal lRow As Long, ByVal lCol As Long)
   If iGrid1.RowIsGroup(lRow) Then
      Dim i As Long
      For i = 1 To iGrid1.RowLevel(lRow) + 1
         Debug.Print "Level " & i, _
            iGrid1.CellValue(lRow, iGrid1.GroupObject.SortCol(i))
      Next
   End If
End Sub
```

3. [Fixed] The **GroupObject** is cleared after the grid has been ungrouped interactively or in code using the **Ungroup** method as the **GroupObject** should reflect the current grouping state of the grid.

4. [Fixed] In some rare cases iGrid might crash the app while the mouse pointer is moving inside its column headers.

5. [Fixed] For some TrueType fonts the auto-width operation might work incorrectly – the column width wasn't enough to display the column header and cell texts without clipping.

6. [Fixed] iGrid might crash the app after you had displayed the default column header context menu and had cancelled it by clicking somewhere in the cell-free area.

7. [Fixed] Grouping was case-sensitive if you used the **igSortByCellTextNoCase** sort type.

8. [Fixed] If iGrid had a tree with an invalid hierarchy structure and you sorted the grid by this column, iGrid might crash the VB IDE. Now a message box with the corresponding error text is displayed in this case and the grid remains stable.

9. [Fixed] You might need to hit DOWN ARROW twice to move from a row to the next row when it had row text cell in row mode (the same concerns UP ARROW).

10. [Fixed] The **RemoveRow** method might trigger an error when you tried to remove the last row or some last rows.

11. [Fixed] iGrid might trigger an error when one of its group rows was selected and you applied another grouping.

12. [Fixed] The **TextEditChange** event wasn't raised after the user had pressed an alpha-numeric key to start the text editing.