10Tec iGrid ActiveX 5.0
What's New in the Release

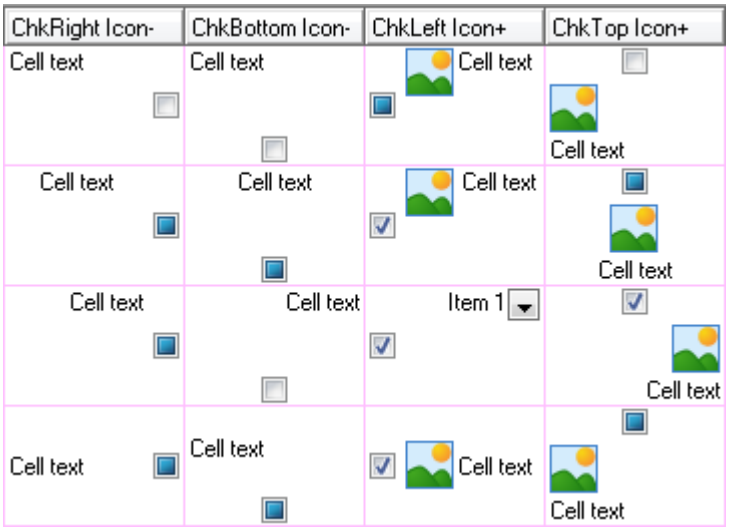
Table of Contents

Keywords used to classify changes

- [New] – a totally new feature;
- [Change] – changes in a member functionality or interactive behavior;
- [Improvement] – a feature works better than in a previous version;
- [Removed] – a feature was totally removed;
- [Renaming] – a name of the member was changed so it is enough to rename it in your code.

New Cell Check Box System

1. [New][Removed] This version of iGrid introduces the ability to have a check box control in a cell of any type in addition to the standard cell contents (such as icons and text). Previous builds of iGrid only allowed check box cells without additional contents as a special cell type (the **igCellCheck** flag from the **ECellTypes** enumeration), but now you can optionally display check box in any cell and access/set the check box state value as a separate value, independent of the main cell value:



To display a check box in a cell, use the new **CellCheckVisible** Boolean property. Just set it to True to display check box in addition to the standard cell contents (extra icon, main icon, text, etc). Note that even combo box cells can now have the check box control.

The new check box system makes unneeded the old check box cell type and the corresponding **igCellCheck** item in the **ECellTypes** enumeration – they are no longer present in iGrid.

2. [New] The cell check box can be placed in one of the possible 5 positions in a cell: at the top, the bottom, the left, the right, or in the middle of the cell (i.e. centered vertically and horizontally). In the first 4 positions it is also centered near the corresponding cell edge.

   The new **CellCheckPos** property is used to specify check box placement. It accepts one of the values from the new **ECellCheckPos** enumeration:

```
Public Enum ECellCheckPos
   igCheckPosLeft = 0
   igCheckPosTop = 1
   igCheckPosRight = 2
   igCheckPosBottom = 3
   igCheckPosCenter = 4
End Enum
```

   By default, the cell check box is displayed at the left cell's edge (**igCheckPosLeft**).

3. If a cell contains a check box, iGrid draws it first before the rest of the cell contents are drawn. It then makes an indent in the corresponding direction to the cell contents unless the check box is placed at the center of the cell. This indent between the check box and the main cell contents is controlled with the new **CellCheckGap** Long property. It works similar to the **CellIconGap** property for the cell icons. The default value for **CellCheckGap** is 2 pixels.

4. The state of a cell check box can be read/set using the new **CellCheckState** property. It contains one of the possible 3 states defined in the new enumeration **ECellCheckState**:

```
Public Enum ECellCheckState
   igCheckStateUnchecked = 0
   igCheckStateChecked = 1
   igCheckStateGrayed = 2
End Enum
```

5. [New] The **CellObject** has new properties corresponding to the following 3 new cell properties used to control cell check boxes: **CellCheckVisible**, **CellCheckState**, and **CellCheckPos**. These are **bCheckVisible**, **eCheckState, eCheckPos** respectively.

6. [New][Change] When you try to change the state of a cell check box using visual interface (say, with the mouse click), the new event **CellCheckChange** is fired:

```
Event CellCheckChange(ByVal lRow As Long, ByVal lCol As Long, _
   ByRef eNewCheckState As ECellCheckState, ByRef bCancel As Boolean)
```

   It's fired just before the check state changes and allows you to cancel the forthcoming change. It's done with the help of the **bCancel** parameter passed by reference. To do that, simply change its default value from False to True.

   The event also lets you read the check state through the **eNewCheckState** parameter and change it on the fly as needed.

   Note that the check box state isn't bound to the cell value anymore, so the **RequestEdit** event isn't fired when changing the check box state. You should use the **CellCheckChange** event instead.

7. [New][Change] To change the check state of the current cell using the keyboard, the SPACE key is used. This key no longer starts cell editing.
8. [New] Two new sort types, **igSortByCheckState** and **igSortByCheckVisible** (from the **ESortTypes** enumeration), were implemented. The first of them allows you to sort a column by the states of the cell check boxes in it in the order unchecked/checked/grayed. The second sort type can be used if you need to sort a column containing cells with and without check boxes inside; in this case, after sorting, all the cells without check boxes will be placed before the cells with check boxes.
9. [New] When grouping the grid using the new **igSortByCheckState** sort criteria, the group rows have the following values: "Unchecked", "Checked", "Grayed". These default strings can be changed through the **UIStrings(26)** - **UIStrings(28)** calls.

   The same concerns the **igSortByCheckVisible** sorting: "Checkbox hidden" and "Checkbox visible" are displayed in this case, and they can be localized using the **UIStrings(29)** and **UIStrings(30)** values.
10. [New] The **Editable** property also now affects cell check boxes. If it is set to False, you can neither edit cell text nor change cell check box state interactively.
11. [Change] **ECellTypeFlags**: the numeric values for flags were changed because the **igCheckBoxFlat** and **igCheckBox3State** flags can be used now with cells of any types. No changes are required in your code if you used the named strings instead of the flag numeric values.
12. [Change] The **FillFromRS** method was adapted to use the new cell check box system. If you called **FillFromRS** for an empty iGrid, now the method creates columns with **CellCheckVisible** set to True for the Boolean fields and no changes are required in your code. If you defined your own column set using the **AddCol** methods before you populate the grid with **FillFromRS**, just set **CellCheckVisible** to True for the columns you wish to display check boxes in.

Guide to upgrade your code from earlier versions

- Previously you'd set **CellType** to **igCellCheck** to have check box cells. You should now set **CellCheckVisible** to True instead.

Old code:

```
iGrid.AddCol().eType = igCellCheck
```

New code:

```
iGrid.AddCol().bCheckVisible = True
```

- If you needed to position the cell check box control, you used the **CellAlignH**/**CellAlignV** properties. This has been replaced with the call to **CellCheckPos**. For instance, if you set **CellAlignH** to **igAlignHCenter** to center the check box control using the default vertical alignment (top), in iGrid 5.0 you set **CellCheckPos** to **igCheckPosTop** for that.

Old code:

```
With iGrid.AddCol()
   .eCellAlignH = igAlignHCenter
   .eCellAlignV = igAlignVTop
End With
```

New code:

```
iGrid.AddCol().eCheckPos = igCheckPosTop
```

- The **CellValue** property no longer controls the cell check box state. To set the check state, assign one of the **ECellCheckState** values to the **CellCheckState** property.

Old code:

New code:

```
With iGrid.AddCol()
   .eType = igCellCheck
   .vValue = True
End With
```

```
With iGrid.AddCol()
   .bCheckVisible = True
   .eCheckState = igCheckStateChecked
End With
```

- In the previous versions you reacted to check box state changes using such editing-related events as **BeforeCommitEdit**; you should now use the new **CellCheckChange** event instead.
- To sort the columns with check boxes by their values, you must specify the **igSortByCheckState** sort type for the corresponding columns explicitly.

Old code:

```
iGrid.AddCol()
```

New code:

```
iGrid.AddCol(eSortType:=igSortByCheckState)
```

Full Unicode Support in Members

1. [New] iGrid implements the new Boolean read-only property **IsUnicode** which can be used to know whether iGrid fully supports Unicode editing and displaying of characters (i.e. iGrid works in an NT-kernel OS).
2. [New][Change] Latest versions of iGrid support displaying and editing of Unicode characters on all modern Windows systems based on the NT kernel, but the key code parameters of some methods and events were still ASCII for backward compatibility with old VB6/VBA coding standards. This release of iGrid introduces full support of Unicode character codes in the members. The changes were made in the following members:
   - The Integer **KeyAscii** parameter of the **KeyPress** and **TextEditKeyPress** events was replaced with the **CharCode** parameter of the Long data type.
   - The Integer **iKeyAscii** parameter of the **RequestEdit** event was replaced with the **lCharCode** parameter of the Long data type.
   - The Integer **iKeyAscii** parameter of the **RequestEditCurCell** method was replaced with the Long **lCharCode** parameter.

   if iGrid works in a Windows NT-based system that fully supports Unicode (the **IsUnicode** property returns True), all the **CharCode** parameters listed above contain the Unicode code of the character. If iGrid works in a Win9x system (**IsUnicode** is False), the **CharCode** parameters still contain the ASCII character code as if you used the old **KeyAscii** parameter.

Guide to upgrade your code from earlier versions

All existing code will work without changes if you deal with the standard ASCII characters with codes from 0 to 127. If you used non-English characters and symbols with codes from 128 to 255, you will need to use the equivalent Unicode character codes on Windows NT systems. The corresponding type conversion and string functions (such as ChrW/AscW) should be used instead of ASCII functions in this case.

If you used code like this:

```
Private Sub iGrid1_TextEditKeyPress(ByVal lRow As Long, ByVal lCol As Long,
KeyAscii As Integer)

   Dim sChar As String

   sChar = Chr(KeyAscii)

   ' Doing something with sChar...
End Sub
```

, now it should be converted to the following one for the general case:

```
Private Sub iGrid1_TextEditKeyPress(ByVal lRow As Long, ByVal lCol As Long,
CharCode As Long)

   Dim sChar As String

   If iGrid1.IsUnicode Then
      sChar = ChrW(CharCode)
   Else
      sChar = Chr(CharCode)
   End If

   ' Doing something with sChar...
End Sub
```

(Pay attention to the fact that the Integer **KeyAscii** parameter was also changed to **CharCode** as Long).

But if you take into account the fact that your app will work only in new OS's like Windows XP or higher which all are Unicode systems, you can simplify your code to the following one:

```
Private Sub iGrid1_TextEditKeyPress(ByVal lRow As Long, ByVal lCol As Long,
CharCode As Long)

   Dim sChar As String

   sChar = ChrW(CharCode)

   ' Doing something with sChar...
End Sub
```

Other Changes and New Features

1. [New] The new **AfterCellDraw** event was implemented:

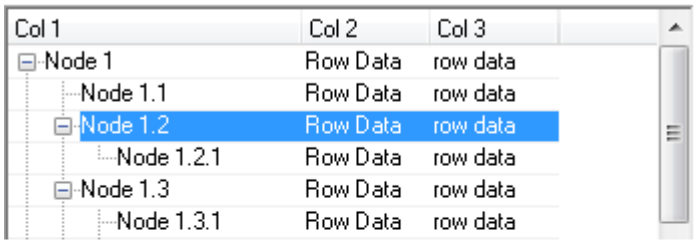
   ```
   Event AfterCellDraw( _
      ByVal lRow As Long, ByVal lCol As Long, ByVal hdc As Long, _
      ByVal lLeft As Long, ByVal lTop As Long, _
      ByVal lRight As Long, ByVal lBottom As Long, _
      ByVal bSelected As Boolean)
   ```

   If we compare this event with the **BeforeCellContentsDraw**, **AfterCellContentsDraw** or **CustomDrawCell** event, it's raised after all drawings in a cell have been finished (i.e. focus rectangle, grid lines, tree button control, etc have been drawn). This allows you to draw on top of all other cell items, which can be useful if you need to place special status icons over the standard cell contents, for example.

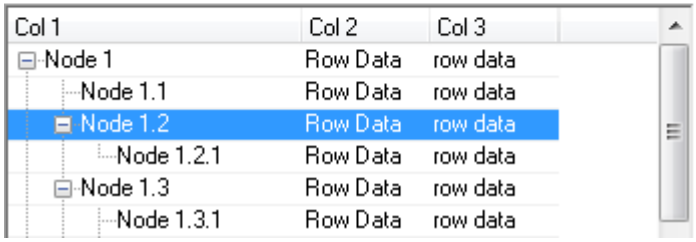
2. [New] The Boolean **FullRowSelect** property was implemented to allow the entire row to be selected rather than starting at the indent level. By default, if you have a tree in the first visible column of iGrid,

the level indent area isn't selected when row mode is on. This behavior corresponds to the default value of this property, i.e. False. To select the whole row starting from the left edge of the grid, set this property to True. The following picture illustrates this property:

FullRowSelect = False:



FullRowSelect = True:



3. [Change] In previous versions iGrid could be scrolled horizontally using the mouse wheel together with the CTRL key. This combination became a "de facto" standard to zoom documents and worksheets in web browsers and grid applications like MS Excel. Because of this CTRL+MOUSEWHEEL is no longer used in iGrid at all so the developer can implement its own action for this combination. Instead, SHIFT+MOUSEWHEEL is now used to scroll the grid horizontally.

4. [New] The new **MouseWheel** event to process mouse wheel input was implemented:

```
Event MouseWheel( _
   ByVal lDistance As Long, _
   ByVal Button As Integer, ByRef Shift As Integer, _
   ByVal x As Single, ByVal y As Single, _
   ByRef bDoDefault As Boolean)
```

The **lDistance** parameter gives you information about the current mouse wheel rotation. A positive value indicates that the wheel was rotated forward, away from the user; a negative value indicates that the wheel was rotated backward, toward the user. The absolute value equals the number of virtual rows scrolled by the wheel one notch (can be viewed/changed in the OS Control Panel, see the Mouse Properties applet, the Wheel tab, the Vertical Scrolling group).

Button, **Shift**, **x**, **y** are the standard parameters of mouse events which provide you with information about what mouse buttons and special keys (such as ALT, CTRL) were pressed, and the mouse pointer position. Note that the **Shift** parameter is passed by reference. This allows you to change the built-in mouse wheel scrolling. For instance, if you assign the **vbShiftMask** constant (1) when **Shift** is **vbCtrlMask** (2), iGrid will be scrolled horizontally by both CTRL+MOUSEWHEEL and SHIFT+MOUSEWHEEL.

The **bDoDefault** parameter is used if you wish to prohibit the built-in scrolling when the user rolls the mouse wheel and implement your own action. To do that, set this parameter to False and implement your own action in this event.

The **ScrollBarBeforeScroll** event also allows you to detect mouse wheel rotation using its **bMouseWheel** parameter, but this event, in contrast to **MouseWheel**, is raised only if iGrid has the corresponding

scroll bar and iGrid is about to scroll its contents as the result of mouse wheel rotation. **MouseWheel** is raised in any case when the wheel is rolled by the user, even if iGrid has no scroll bars.

5. [New] The **AutoWidthCol** method was supplemented with the new optional **lMinimumWidth** parameter you use to specify the minimum column width during the auto-sizing operation.
6. [New] The new **igTextWordEllipsis** flag was added to the **ETextFormatFlags** enumeration. This flag corresponds the DT_WORD_ELLIPSIS flag used in the WinAPI DrawText function and is used to truncate each line of multi-line cell text with ellipsis if such a line exceeds the cell width.
7. [New] The Boolean **BuiltinContextMenus** property was implemented. By default its True, but if it is set to False, iGrid never displays any built-in context menus. These are the cell context menu with copy/paste and optional collapse/expand commands (if you click a tree node or a group row), and the column header context menu with the commands used to sort and group/ungroup iGrid.
8. [Improvement] In the previous versions, when you grouped the grid using the **igSortBySelected** criteria, the False or True words were placed in the group rows as the group values by default. This version displays more meaningful terms "Non-selected" and "Selected", and they can be also access/changed (localized) through the **UIStrings(17)** and **UIStrings(18)** calls respectively.

   The same concerns the **igSortByHatch** sort type. The previous version displayed 0-5 or 7 for the corresponding hatch types from the **EHatchStyles** enumeration, but now the following words are displayed: "Horizontal hatch", "Vertical hatch", "45-degree downward hatch", "45-degree upward hatch", "Horizontal and vertical crosshatch", "45-degree crosshatch", "No hatch". These string values are controlled through the **UIStrings(19)** – **UIStrings(25)** calls respectively.
9. [Improvement] The drawing code of iGrid was optimized in the latest version, therefore cell contents are drawn faster than previous versions.
10. [Improvement] iGrid no longer draws focus rectangle around the selected item in the combo lists to make the interface clearer.
11. [Change] iGrid caches all used format strings (specified through the **CellFmtString** property), and the previous version allowed you to use a ridiculous number (32767) of different format strings in one iGrid. The current version of iGrid allows you to use up to 255 different format strings enough for the vast majority of cases.

    The fact is that iGrid stores all cell properties in a highly efficient bit representation to minimize memory usage (just 48 bytes to store all cell properties). The internal structures were repacked and redesigned to accommodate the new cell check properties. The room for the new properties was taken from such capabilities as a superfluous (in real-world apps) number of potential format strings iGrid can cache inside.
12. [Change] The cell check box and combo button controls aren't scaled in this version if there isn’t enough space to display them full size. In earlier versions of iGrid you may see smaller cell controls in this case, and their sizes can differ in various cells. This didn’t look as nice as the current implementation when using controls of the same size, and they are just cropped if there is no enough space to display them fully.
13. [Change] The order and corresponding numeric values of the constants from the **ESortTypes** enumeration were changed because of the new (**igSortByCheckState, igSortByCheckVisible**) and removed (**igSortByIndentation**) items. This change affects your code only if you use numeric values instead of named values (this often happens in non-VB or VBA based development environments).

14. [Removed] The almost unused **igSortByIndentation** sort type (the **ESortTypes** enumeration), which was used to sort the grid by the left cell indent, was removed. If you need to sort the grid by the left or another internal contents indent, you can use custom sorting (the **igSortCustom** sort type).

Fixed Bugs

1. [Fixed] A serious bug with the Copy/Cut/Paste commands applied to the entire control in design mode in MS Access was fixed. The bug was present in all 4.x versions of the control. When you tried to copy/cut the control into the clipboard, the run-time error 97 "Can not call friend function on object which is not an instance of defining class" occurred, and MS Access stopped responding. However, it was possible to copy/paste an iGrid instance with other controls, but on some step of development you might get other error messages, such as "There is no object in this control" or "The expression On Load you entered as the event property setting produced the following error: Sub or Function not defined".

   The nature of this problem is the following. MS Access has problems when accessing the unified Ambient property for all ActiveX controls when doing clipboard operations. This property is used in the initialization code of iGrid to access some standard control container's properties, such as Font and UserMode. This technique works well in other development environments, such as VB6 and even MS Word/Excel VBA, but it fails in MS Access and only when doing copy/paste.

1. [Fixed] An important problem with internal row key index corruption was fixed. iGrid might lose row keys after calling the **RemoveRow** method.
2. [Fixed] A serious bug causing tree lines to remain visible when sizing the tree column.
3. [Fixed] A serious problem with entering non-English characters was fixed. In previous versions, if you pressed a non-English character to start editing, another symbol was placed into the cell editor. On some systems (mainly on Windows XP), the corresponding **KeyPress** event might be triggered twice with improper values for the **KeyAscii** parameter in this case.
4. [Fixed] Assigning large values to the **ColPos** property (which is greater than the total number of columns) didn't raise an internal iGrid error and might have led to improper grid behavior.
5. [Fixed] When you pressed the right mouse button when the pointer is over a cell, a control in the cell could be affected (the cell's check box state could be changed, or the cell's combo box button might be pressed). Now you can affect any cell control using only the left mouse button. The right mouse button just selects the cell and displays the default context menu.
6. [Fixed] iGrid might crash your application if you pressed ALT+F4 to close the application when the input focus was in iGrid and a combo box cell was selected.
7. [Fixed] If a cell contained the CR/LF characters, the built-in cell tooltip was shown improperly – the CR/LF characters might be ignored, or the text was truncated at the first occurrence of CR/LF. The same concerns the **RequestCellToolTip** and **RequestColHeaderToolTip** events: if you used vbCrLf when making your custom tooltip text in the **sTipText** parameter, you encountered the same problem.
8. [Fixed] In row mode, if you had set the **CurRow** property to a positive value when there was no current cell (**CurRow** and **CurCol** were 0), iGrid crashed your app when you pressed the UP ARROW or DOWN ARROW key.
9. [Fixed] A series of problems with displaying corresponding help topics for some iGrid members when F1 was pressed in the editor were fixed.
10. [Fixed] iGrid might be scrolled by the mouse wheel rotation when the **MouseWheelScrollsWhenHidden** property of the corresponding scroll bar object was set to True and iGrid did not require the scroll bar.

11. [Fixed] Editing started on double-click done with the right mouse button when the **CopyPasteEnabled** property was set to True.
12. [Fixed] iGrid might be scrolled and the current cell was changed to an unpredictable cell after PAGE UP/PAGE DOWN had been pressed when there were non-selectable cells after/before the current cell.
13. [Fixed] iGrid did not raise the **TextEditKeyPress** event for the characters with codes 33-40 /these are !"#$%&'(/.
14. [Fixed] iGrid crashed on the Copy or Cut command after the **SelectAll** method had been issued.
15. [Fixed] The **SelectAll** method always marked the row text cells as selected regardless their usage status provided by the **DrawRowText** property.
16. [Fixed] Some bugs with copying/pasting row text cells.
17. [Fixed] iGrid did not start editing when pressing ENTER or doing mouse click/double-click when the **KeyPressBehaviour** property was set to **igKeyPressSearchCurCol** or **igKeyPressSearchDefCol**.
18. [Fixed] In Windows Vista and Windows 7, when your exe application used OS visual styles, iGrid didn't highlight the selected item in the drop-down list in a combo box cell when you opened the list.