10Tec iGrid ActiveX 5.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.

v5.00, build 0105 | 2014-Dec-11

1. [Enhancement] In virtual mode (the **Virtual** property is True), if the user tries to paste data into iGrid and there is not enough space for the data to paste, iGrid automatically adds as many rows as required instead of displaying the corresponding confirmation. The **RequestRow** event related to virtual mode is also raised at that, allowing you to restrict the number of rows to add if required. Note that this enhancement does not affect the column set: new required columns aren't added automatically in virtual mode.

   The first line of the text in the confirmation was changed from "Note enough space to paste all data." to "Note enough rows/columns to paste all data." to avoid any ambiguity for the user (not enough memory, etc). As in the previous builds, this string can be changed/localized using the **UIStrings(16)** property of iGrid.

   In the previous builds, the string "10Tec iGrid Control" was displayed in the title of the confirmation box. In this version, this string was changed to "Paste operation", and it can be changed/localized using the new **UIStrings(31)** property of iGrid.

2. [Fixed] Calling **EndUpdate** without the corresponding prior **BeginUpdate** call may have caused problems with redrawing the grid contents.

v5.00, build 0102 | 2014-Oct-28

1. [Enhancement] Many iGrid events provide you with the **lRow** and **lCol** parameters, which give you information about the cell (row, column) related to the event. In some of these cases **lRow**/**lCol** are always non-zero and represent a real cell because the event could not be triggered without this cell. For instance, it is the **AfterCommitEdit** event, which is raised only if a cell has been edited. In other cases **lRow** and/or **lCol** can be zero as there is no cell related to the event. This is true for such events like **MouseDown** or **Click** when the user presses the mouse button on the empty space in iGrid. The full list of these events is the following:

- `CurCellChange(ByVal lRow As Long, ByVal lCol As Long)`
- `Click(ByVal lRow As Long, ByVal lCol As Long)`
- `ClickNoDblClick(ByVal lRow As Long, ByVal lCol As Long)`
- `DblClick(ByVal Button As Integer, ByVal Shift As Integer, ByVal x As Single, ByVal y As Single, ByVal lRow As Long, ByVal lCol As Long, ByRef eAction As EDblClickAction)`
- `HeaderRightClick(ByVal lCol As Long, ByVal Shift As Integer, ByVal x As Long, ByVal y As Long, ByRef bDoDefault As Boolean)`
- `MouseDown(ByRef Button As Integer, ByRef Shift As Integer, ByVal x As Single, ByVal y As Single, ByVal lRow As Long, ByVal lCol As Long, ByVal eCellPart As ECellParts, ByRef bDoDefault As Boolean)`
- `MouseMove(ByVal Button As Integer, ByVal Shift As Integer, ByVal x As Single, ByVal y As Single, ByVal lRow As Long, ByVal lCol As Long)`
- `MouseUp(ByRef Button As Integer, ByRef Shift As Integer, ByVal x As Single, ByVal y As Single, ByVal lRow As Long, ByVal lCol As Long, ByRef bDoDefault As Boolean)`
- `BeforeRowCollapseExpand(ByVal lRow As Long, ByVal bNowExpanded As Boolean, ByRef bDoDefault As Boolean)`
- `AfterRowCollapseExpand(ByVal lRow As Long, ByVal bNowExpanded As Boolean)`

To help the developer to know whether these parameters can be zero and they should be checked for that, they were renamed to **lRowIfAny** and **lColIfAny** in all the signatures of the events listed above.

Note that due to the COM binary compatibility rules, when only parameter types and their order matters, you do not need to rewrite your code or recompile existing apps.

2. [Enhancement][Change] The **BeforeCollapseExpand** and **AfterCollapseExpand** events are now raised when the user collapses/expands tree nodes using the following commands from the built-in cell context menu: Collapse All Rows, Expand All Rows, Collapse All Child Rows, Expand All Child Rows. However, in all these cases the **BeforeCollapseExpand** and **AfterCollapseExpand** events are raised just one time. If the users selects the Collapse All Child Rows or Expand All Child Rows command, the **lRowIfAny** parameter of the events contains the index of the node the command is applied to (similar to the case when the NUMPAD * key is used to fully expand a tree node). If the user selects Collapse All Rows or Expand All Rows command, **lRowIfAny** equals 0 that indicates this is a global operation for all grid rows.

   Raising these events just one time but not for every row involved into the operation saves the CPU resources and allows the developer to react just one time to one user action (for instance, automatically adjust the width of the tree column after expanding a series of nodes as one operation).

   IMPORTANT NOTE: The **lRowIfAny** parameter of the **BeforeCollapseExpand**/**AfterCollapseExpand** events can be equal zero now, so you need to check this parameter value in your event handlers if you use it to access grid rows.

3. [Enhancement][Change] If a tree node is removed with the **RemoveRow** method, the tree button in the parent node is automatically hidden if the parent node does not have child rows after the removal. In other words, the **RowTreeButton** for the parent row is set to **igRowTreeButtonHidden** if the previous value was **igRowTreeButtonVisible** in this scenario. The same logic is applicable to group rows and their child rows as they are based on the same row level ideology.

4. [Enhancement] The NUMPAD * key now expands all child rows of the current tree node or group row even if it has been already expanded (in the previous builds, it was done only if the current node or group row was collapsed).
5. [Fixed] Tree lines were drawn incorrectly for some combinations of tree nodes.
6. [Fixed] The **Sys(igSysRowsVisScrollCount)** call returned improper values in the following cases:
   - after the user had expanded a tree node using the NUMPAD * key;
   - the **RowVisibleAsChild** property had been changed from code;
   - after calling the **MoveRow** method in some cases.
7. [Fixed] A row became corrupted after assigning True to its **RowVisibleAsChild** property.
8. [Fixed] In some cases iGrid was drawn incorrectly if the user had used the NUMPAD * key to fully expand a tree node.
9. [Fixed] If the user collapsed a tree node so the vertical scroll bar must have disappeared, the image of the scroll bar remained visible in the grid area.
10. [Fixed] Some drawing artifacts appeared near the vertical scroll bar if the **Appearance** property was set to **igAppearanceFlat**.

v5.00, build 0094 | 2014-Aug-21

1. [New] Two new keyboard combinations can be used during incremental search: ALT + DOWN ARROW finds the next match below the current cell, ALT + UP ARROW finds the previous match above the current cell. If there is no next match when pressing these keys, the current cell remains unchanged.
2. [Optimization] Incremental search works much faster when multiselection mode is on.
3. [Fixed] In multiselection mode, the **BeforeSelectionChange** and **AfterSelectionChange** events were not raised when the current cell or row was changed during incremental search.
4. [Fixed] New cells added to the grid were marked as selected after the user had pressed the DEL key or the CTRL+X key combination.

v5.00, build 0090 | 2014-Apr-11

1. [Fixed] The settings made through the **eTextEditOpts** parameter of the **RequestEdit** event did not affect the editing process.
2. [Fixed] The leftmost vertical grid line of normal rows, displayed as children rows of group rows, were not drawn correctly when the grid contents were scrolled horizontally.

v5.00, build 0086 | 2013-Dec-18

1. [Enhancement] If you specified the **igTextWordBreak** flag in the **CellTextFlags** property to enable word wrapping when cell text is displayed, this flag was not used while editing cells. This build of iGrid inherits this setting in edit mode.
2. [Enhancement] The **FindSearchMatchRow** method checks the correctness of the optional **lStartRow** parameter. If an incorrect row number is passed, the "Invalid procedure call or argument" error is generated. In the previous versions, the method used the value 1 in this case which could be misleading.
3. [Enhancement] The **oFont** property of the column default cell object is initialized using the copy of the current grid font when a column is created (Nothing was used in the previous versions). This allows the developer to modify the font for new cells in a column easily if a font similar to the grid font should be

used, without writing many statements to clone the grid font object. For instance, now the developer can use the following statement to make the contents of all new cells italic in the newly created column:

```
iGrid1.AddCol.oFont.Italic = True
```

(the **AddCol** method returns a reference to the column's default cell object).

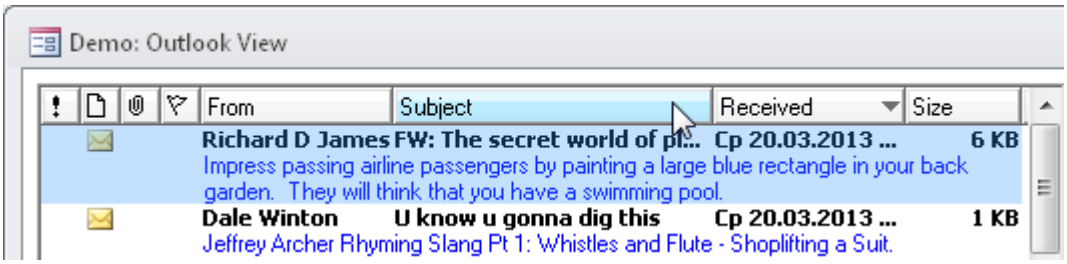
4. [Enhancement] Some internal routines related to font processing were optimized. Now iGrid uses fewer font resources if the default cell font was changed and then set back to the default grid font.
5. [Fixed] The displayed cell text was truncated to 255 characters if the cell format string (the **CellFmtString** property) had been set to an empty string.

v5.00, build 0081 | 2013-May-17

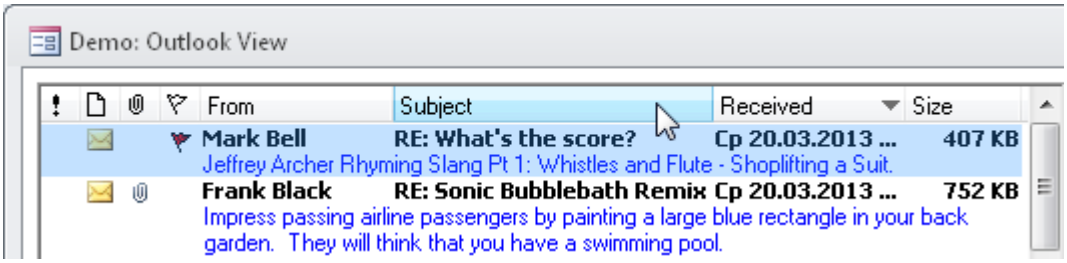
1. [Fixed] A series of problems with the built-in Paste command when the clipboard was empty were fixed. The Paste command was available in the built-in context menu or by the CTRL+V keyboard combination in this situation, and issuing it caused the crash of the entire application. If the Paste command was invoked while editing a text cell, the selected text was simply deleted.
2. [Fixed] A serious bug in the **RemoveRow** method was fixed. If the grid had selected cells or rows, changing the current cell might crash the entire application after calling this method. The **SelItems** object might also return improper selected object collection after the call to **RemoveRow**.
3. [Fixed] The column header borders might disappear if the column header contents were wide than the available column header rectangle.

v5.00, build 0078 | 2013-Mar-21

1. [Enhancement] The iGrid header drawing routine was rewritten to use the OS visual style in Windows Vista or later systems, including Windows 7 and Windows 8, without any additional efforts. Earlier the developer should have incorporated a special manifest into the application to enable this feature, which is totally impossible for the MS Office VBA development. Now all users of iGrid in MS Access, Word and Excel can see the styled header instead of the old 3D look. Compare the old header



to the new one we see in MS Access without any recoding:



As always, the automatic use of the new look can be turned off using the **iGrid.Header.UseXPStyles** property.

2. [Fixed] Some minor drawing errors in the header were fixed.

3. [Fixed] The contents of the selected row text cells aren't placed into the clipboard when using the built-in copy/cut command.

4. [Fixed] Changing a column header icon through the **ColHeadeIcon** property might cause internal data structures corruption and improper cell drawing (as if the corresponding column width were set to 0 while the column header remained visible).

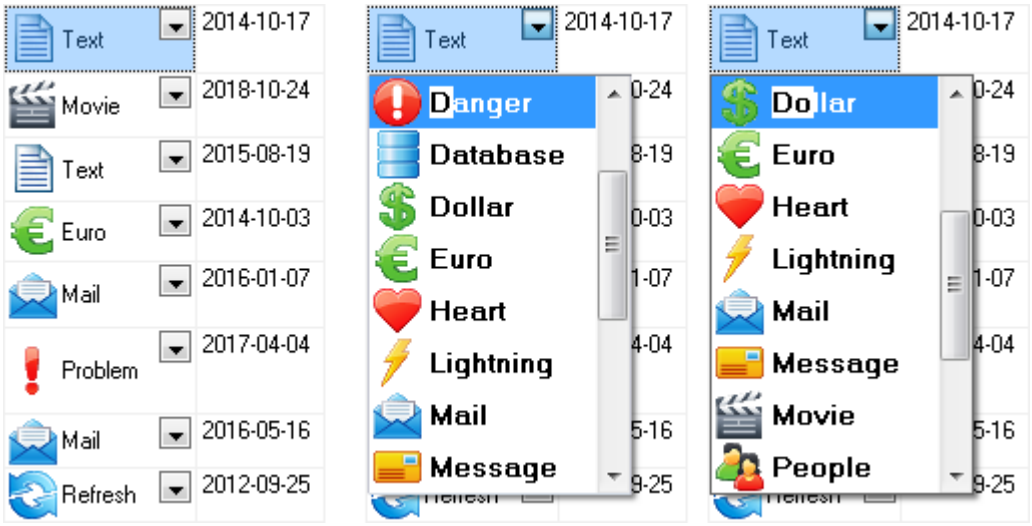
v5.00, build 0074 | 2012-Oct-12

1. [Enhancement] The incremental search in combo lists was implemented instead of the ability to find the required item only by the first letter.

   If a drop-down list is opened, type the first characters of the item you want to find. iGrid will select the first item that matches your search if it exists. To help you see what characters have been pressed, the first characters of the found item will be drawn using the inverse selection colors.

   To make this process even faster, iGrid automatically activates the incremental search mode in a combo box cell when you start editing by pressing an alpha-numeric key. If there is an item that starts with the pressed character, iGrid opens the drop-down list and automatically highlights this item.

   Here is a series of screenshots of what's happening in an iGrid combo box cell when iGrid was not in edit mode and the keys 'D' and 'O' were pressed:



   The items in the drop-down list may be not sorted alphabetically. iGrid always searches for the item testing all the items from the top to bottom one-by-one, and thus allows you to find the required item regardless of item order. If there are several items which start with the specified characters, you can always move from one of these items to another using the CTRL+UP ARROW and CTRL+DOWN ARROW combinations.

   Your incremental search is always reset if you select another item by placing the mouse pointer over it or if you select it using the caret move keys on the keyboard - such as UP ARROW, PAGE DOWN, etc. (except the special CTRL+UP ARROW and CTRL+DOWN ARROW combinations described above).

2. [Fixed] The cell check boxes were drawn disabled when the **Enabled** property had been set to False in the Property Grid and then changed to True in code.

3. [Fixed] If the **MultiSelect** property was set to True, iGrid selected cells/rows when moving the mouse pointer as if drag select mode were in effect after clicking a combo box button and closing the drop-down list.

v5.00, build 0068 | 2012-May-07

1. [Enhancement] The algorithm that draws the cell contents with the **igTextSingleLine** and **igTextPathEllipsis** text format flags was enhanced. Now you see more useful characters instead of slashes and ellipsis if the cell width is not enough to display the full text.
2. [Fixed] iGrid could crash when drawing cell texts with slashes if the **igTextSingleLine** and **igTextPathEllipsis** text format flags used in the same cell.
3. [Fixed] If the only **igTextSingleLine** text format flag was specified for a cell, it acted as if the **igTextPathEllipsis** flag were also in effect and the cell text was truncated at the middle using ellipsis when the text had slashes ("\" or "/").
4. [Fixed] An application with iGrid could hang because of infinite context switching between the app and the Remote Desktop Clipboard Monitor process in Windows (rdpclip.exe). The problem was related to the interaction with the Windows clipboard using the clipboard viewer chain method which had come from old versions of Windows.

v5.00, build 0062 | 2012-Feb-03

1. [Enhancement] The core of the built-in copy/paste functionality was highly optimized. In earlier builds, copy and cut operations could take much longer when copying arrays of data into the clipboard (1000 cells or more). The optimization in this build allows these operations to perform up to 100 times faster!
2. [Enhancement] The drawing code was significantly optimized. Now iGrid may take much less CPU resources (up to 20 times) when changing any Cell* property - such as **CellValue**, **CellBackColor**, etc - for the cells in the rows which are not visible in the viewport of iGrid when redrawing is on. The optimization also concerns the rows with the **Visible** property set to False and collapsed rows nested into other rows (sub-group rows, tree nodes in other collapsed nodes, etc).
3. [Fixed] A critical bug you could encounter while working with iGrid mainly at design-time was fixed. It appeared as the message box with the "System Error &H80043553 (-2147207853)" and/or "Out of memory" messages after you opened/closed or launched your form with iGrid many times.
4. [Fixed] The cells of last rows became corrupted when issuing **AddCol** and/or **RemoveCol** methods after removing the first row(s) with the **RemoveRow** method.
5. [Fixed] The **Clear** method did not erase the internal counter of visible scrollable rows you can retrieve with the **iGrid.Sys(igSysRowsVisScrollCount)** call.
6. [Fixed] iGrid did not update a cell on the screen when the **CellCtrlKey** property was changed.

v5.00, build 0052 | 2011-Dec-12

1. [Fixed] In MS Access, iGrid may have frozen the entire development environment when you double-clicked it at design time.
2. [Fixed] In MS Access, iGrid may have hung or its ActiveX container may have become corrupted after some operations in the form layout view.