10Tec iGrid ActiveX 7.0
What's New in the Release

Contents

Tags 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;
- [Code-Upgrade] – a special tag used to mark changes that may require changes in the source code for older versions.

New performance improvement tools

1. [Optimization][Change][New] All iGrid members related to changing row order, height or visibility were optimized a lot. If screen updates are turned off after issuing the **BeginUpdate** method and you call these members one-by-one or in a loop, you may notice a significant performance gain in huge grids. Code snippets changing rows may work up to 200 times faster now, which was true for the following loop in our tests:

```
iGrid1.BeginUpdate

Dim iRow As Long
For iRow = 1 To iGrid1.RowCount
   iGrid1.RowVisible(iRow) = ... ' set row visibility on some condition
Next

iGrid1.EndUpdate
```

   The main properties and methods this performance improvement affects are the followings:

   - Row characteristics - the **RowVisible**, **RowVisibleAsChild** and **RowHeight** properties and the **AutoHeightRow** method.
   - Collapsing/expanding group rows and tree nodes - the **RowExpanded** property and the **CollapseAllChildRows** and **ExpandAllChildRows** methods.
   - Changing row set - the **AddRow**, **MoveRow** and **RemoveRow** methods.
   - Sorting/grouping - the **Sort**, **DoDefaultSort**, **SortRowChildNodes**, **Group**, **Ungroup** methods.

   To implement this optimization, the recalculation of the internal row parameters after every related call is turned off by default. As a result, the correct work of the following members is not guaranteed while updates are turned off:

- The **RowStartY** and **RowVisibleIndex** properties.
- The **CellFromPoint, CellBoundary, EnsureVisibleRow** and **EnsureVisibleCell** methods.
- The **DoKeyboardCommand** call if it leads to scrolling the grid in the vertical direction.
- The **Sys()** call for the **igSysRowsVisScrollHeight** and all **igSysCellsArea*** parameters.

Generally these members are not called while updates are turned off, and the vast majority of apps will not suffer from the new optimization. If you need to provide backward compatibility with existing code and enable normal work of these members, this release of iGrid introduces the new property **BeginUpdateRecalcRowParms**. This Boolean property specifies whether iGrid recalculates the row parameters while updates are turned off. The default value of this property is False. To enable backward compatibility with existing code when members like **RowVisibleIndex** or **CellFromPoint** work correctly while updates are turned off, set the **BeginUpdateRecalcRowParms** property to True.

2. [New] The new **SetColCellsProp** method allows you to effectively set any cell property for all cells or a range of cells in a column. The main benefit of the method is its speed: it may work up to 30 times faster than an equivalent loop enumerating cells in the column like this:

```
Dim iRow As Long
For iRow = 1 To iGrid1.RowCount
   iGrid1.CellCheckState(iRow, 1) = igCheckStateChecked
Next
```

The method signature is the following:

```
Public Sub SetColCellsProp( _
      ByVal vCol As Variant, _
      ByVal eProperty As ECellProperty, _
      ByVal vValue As Variant, _
      Optional ByVal vStartRow As Variant, _
      Optional ByVal vEndRow As Variant _
   )
```

The items of the new **ECellProperty** enumeration are used to specify the cell property you set with this method. Now you can use the statement below to set tick marks in all check boxes in the cells of the first column, instead of the loop above:

```
iGrid1.SetColCellsProp _
   1, igPropCellCheckState, ECellCheckState.igCheckStateChecked
```

The optional **vStartRow** and **vEndRow** parameters allows you to limit the range of cells the method processes. If they are not specified, the specified property is set for all column cells.

3. [New] This release of iGrid allows you to limit the number of rows iGrid scans to adjust the width of a column during the column auto-width operation. This feature can significantly enhance the performance of iGrid during these operations if the user browses only a small set of top rows in huge grids containing thousands of rows.

   This feature is enabled with the new **AutoWidthColMaxRows** property of the Long data type. The property returns/sets the maximal number of rows iGrid should scan when it calculates the optimal column width. The default value of this property is 0, which means that all grid rows are processed.

   The value of the **AutoWidthColMaxRows** property is used when the user adjusts the column width interactively by double-clicking the column divider and when you adjust the column width from code with the **AutoWidthCol**/**AutoWidthCols** methods.

This feature was introduced as an experimental feature in iGrid 6.5.80 and was available through the **DevDbg** call with the parameter index 4. This **DevDbg** call no longer works as the same functionality is provided now by the **AutoWidthColMaxRows** property.

4. [Optimization] The interactive grouping and the **Group** method were optimized to work faster. The performance gain depends on the situation, but the more group rows must be created during grouping, the higher the speed. For example, iGrid works about 3 times faster if it is grouped by a column with unique cell values so that group rows must be created for every value.

5. [Optimization] If you made some last rows invisible, the previous version of iGrid started to work slower in all operations related to changing row parameters. The more rows at the bottom were invisible, the slower iGrid was. This release of iGrid works much faster in this scenario and its performance does not depend on the number of invisible last rows. In the worst case, when all rows are invisible, the new version of iGrid may work 250 times faster compared to its previous version.

6. [Optimization][Removed] To get faster code and a more compact size of the iGrid OCX file, all outdated ANSI Windows API calls related to Windows 95/98 support were removed. Now iGrid can be used only on Windows NT platforms, and the minimal supported version of the OS is Windows 2000. The **IsUnicode** property of iGrid was also removed as it is no longer needed.

Other new features and changes

1. [New][Change] This release of iGrid provides the built-in ability to automatically add columns and rows when the user pastes data from the clipboard if the current set of columns and rows is not enough to accommodate all pasted data. To control this functionality and to notify you of paste operations, iGrid raises two new events - **BeforePaste** and **AfterPaste**.

   The **BeforePaste** event is raised before the paste operation and has the following signature:

```
Event BeforePaste( _
   ByRef bDoDefault As Boolean, _
   ByVal lColsToAdd As Long, _
   ByVal lRowsToAdd As Long, _
   ByRef bAddCols As Boolean, _
   ByRef bAddRows As Boolean, _
   ByRef bShowWarning As Boolean)
```

   The **bDoDefault** parameter passed by reference allows you to cancel the whole paste operation. Its default value is True, which means that the operation is performed.

   The other parameters are used to control the behavior of iGrid when it needs to extend its column and row set to accommodate all pasted data. To know whether iGrid must add columns and rows to accommodate all pasted data, read the values passed in the **lColsToAdd** and **lRowsToAdd** parameters. They contain the number of missing columns and rows to accommodate all pasted data.

   By default iGrid does not add the needed columns and rows. To enable this, set the **bAddCols** and **bAddRows** parameters of the **BeforePaste** event to True. The default value of these parameters is False.

   The last parameter, **bShowWarning**, specifies whether iGrid display its built-in message box telling the user that the current column and row set is not enough to accommodate all pasted data. The default value of this parameter is True, and to suppress this message box, set **bShowWarning** to False in an event handler of the **BeforePaste** event.

   The signature of the **AfterPaste** event raised after the paste operation has been done is the following:

```
Event AfterPaste( _
   ByVal lColsAdded As Long, _
   ByVal lRowsAdded As Long)
```

The **lColsAdded** and **lRowsAdded** parameters of the **AfterPaste** event contain the number of columns and rows respectively added dynamically to iGrid to accommodate the pasted data.

The functionality described above replaces the equivalent functionality from the previous version based on virtual mode. The previous version of iGrid could extend its row set automatically when the current set of rows was not enough to paste all the clipboard data if the **Virtual** property was set to True. Note that this was possible only for rows but not for columns, but now iGrid provides you with the ability to do this for columns too.

2. [New] iGrid raises the following 4 new events related to the copy and cut operations:

```
Public Event BeforeCopy( _
   ByVal bWithHeaders As Boolean, ByRef bDoDefault As Boolean)

Public Event AfterCopy()

Public Event BeforeCut( _
  ByRef bDoDefault As Boolean)

Public Event AfterCut()
```

The **BeforeCopy** and **BeforeCut** events are raised before the copy and cut operations respectively. The Boolean **bDoDefault** parameter of these events passed by reference can be used to cancel the corresponding operation. The **bWithHeaders** parameter of the **BeforeCopy** event also informs you whether cell contents are copied together with column headers.

The **AfterCopy** and **AfterCut** events are raised after the copy and cut operations respectively. If the operation was cancelled in an event handler of the **BeforeCopy**/**BeforeCut** event, the corresponding **AfterCopy**/**AfterCut** event is not raised.

3. [New] The new **Header.HotTrackBackColor** property allows you to set the background color of the active column header under the mouse pointer when the hot tracking effect in the header is on. The default value of this property is CLR_NONE (the hot tracking color is not set).

   To enable the hot tracking effect for column header backgrounds, include the new **igHdrHotBackground** flag from the **EHeaderHotTrackFlags** enumeration in the combination of flags in the **Header.HotTrackFlags** property.

   This feature was introduced as an experimental feature in iGrid 6.5.80 and was available through the **DevDbg** call with the parameter index 5. This **DevDbg** call no longer works as the same functionality is provided now by the **Header.HotTrackBackColor** property.

4. [New] The new Boolean **TreeRootLines** property can be used to turn on or off the drawing of tree lines between root nodes in a tree grid.

5. [New] The **AddRow** method has the new **lNormalCellHeight** parameter. It allows you to specify the height of normal cells above row text cell when you create a new row.

6. [Enhancement][Renaming][Fixed] When you called the **AutoHeightRow** method for a row and the result row height was greater than 32767, the row disappeared from the screen. This problem was related to the limitations of the Integer data type used internally to store row heights.

To eliminate this problem, this version of iGrid uses the Long data type in its internal structures to work with row heights. As a result, the **RowHeight** property also has the Long data type and allows you to create rows with very big values of heights. This feature can come in handy to display big chunks of text or big images, especially on modern high-resolution screens on which the same amount of text or graphical data is rendered using more pixels.

The same changes were done for the **RowNormalCellHeight**, **DefaultRowHeight** and **DefaultRowNormalCellHeight** properties. They all accept/return Long values now.

The type of the **iHeight** parameter in the **SetRowHeights** method was also changed from Integer to Long and the parameter was renamed from **iHeight** to **lHeight** to reflect this change.

7. [Renaming] The **CellCheckChange** event was renamed to **BeforeCellCheckChange** to correspond to the common Before/After event naming system (the corresponding "After" event is **AfterCellCheckChange**).

Fixed bugs

1. [Fixed] When row mode was on, the cells from a new column added with the **AddCol** method were not drawn as selected in the selected row(s).
2. [Fixed] If the last row was invisible and you increased the **RowCount** property to add new rows, their parameters were not set correctly and iGrid drew them improperly on the screen.
3. [Fixed] The **LoadFromArray** method failed for arrays retuned by the **Range().Value** call in Excel VBA.
4. [Fixed] If you called the **AddRow** method and specified its **vRowParent** or **bVisibleAsChild** parameters, the method did not update the internal structures correctly and this led to improper drawing of the grid contents.