10Tec iGrid for .NET
What's New in the Control

v1.70, build 0002

February 8th, 2007

1. [Fixed] When at design-time the developer set the minimal (maximal) width of a column to a value greater (less) than the default column width, an exception was raised at run-time. Now the exception is not raised, but the width becomes equal to the minimal (maximal) width if the specified width was less (greater) than the minimal (maximal) width. The same concerns the behavior of iGrid at run-time. These rules are used only if the corresponding minimal and/or maximal widths were set.
2. [Fixed] The **StartDragCellEvent** was generated multiple times when the user was moving the mouse pointer with the left mouse button pressed over a cell. Now it is raised just once for a cell.
3. [Fixed] When a custom drop-down control was resizable, the height of the drop-down form was set to a value larger than needed.
4. [Fixed] When iGrid was grouped by some columns, the **FitColumnWidths** property of the **iGPrintManager** class worked incorrectly.

v1.70, build 0000 (the release of v1.7)

December 8th, 2006

1. [Change] The ability to implement drag-n-drop operations with several selected cells or rows in iGrid.NET is the main change in this release of the control.

   In the previous versions of the control in a multi-selection mode iGrid.NET changed the selection just after the left mouse button was pressed over:

   - a cell which was previously selected when the Ctrl key was pressed in the multi-extended mode;

   - a cell which was previously selected in the multi-simple mode.

   This behavior put obstacles when you wanted to implement drag-n-drop functionality because when the user pressed the left mouse button over a cell with the aim of dragging, iGrid.NET changed the selection, and you could not prevent it as you did not know whether the user had pressed the mouse button in order to change the selection or to drag the cell (you would know it only when the user would move the mouse pointer with the left mouse button pressed or would release the button). Now in these cases selection is changed only after the mouse button is released.

   A new **DoDefault** parameter was also added to the **CellMouseUp** event. This parameter allows you to prevent a cell from being selected in the cases described above.
2. [New] A new event **StartDragCell** was added. This event allows you to determine the moment when cell dragging should be started. Handle this event if you want to implement a drag-and-drop operation and start dragging when it is raised. A new property named **StartDragCellDelta** allows you specify the distance the mouse pointer should move with a mouse button pressed before dragging is started.
3. [New] A new **FitColWidths** property was added to the **iGPrintManager** class. This property allows you to fit the widths of all the columns to the width of the page when printing.
4. [Fixed] When a check box cell had a value of a numeric type different from Int32, it was processed as a value of unrecognized type; as a result the check box was displayed in the checked state regardless of the value. When the user clicked this check box, the value was changed to False for a two-state check box and to Indeterminate for a three-state check box.
5. [Fixed] When iGrid.NET without columns and a row was focused and the user pressed the Tab or Right Arrow key, the application stopped responding.
6. [Fixed] When iGrid.NET was being added onto a form in the Windows Forms Visual Designer, and this form already had cell styles with names started with the name of the new grid and ended with 'DefaultCellStyle', 'DefaultColHdrStyle', or 'RowTextColCellStyle', the IDE became irresponsible.
7. [Fixed] When the user expanded a group row, and its child row's **Visible** property was set to False, the child row did not become visible even when its **Visible** property was set to True.

v1.60, build 0002

September 8th, 2006

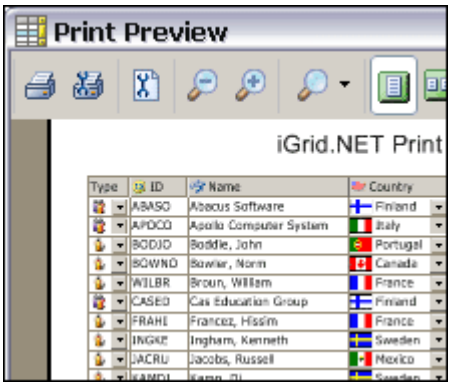
1. [New] Now the iGrid.NET manual is integrated into the help and is available from the Visual Studio help environment.

2. [Fixed] The **FillWithData** method worked incorrectly when the **StaySorted** property was set to True.
3. [Fixed] If a tool tip text had been specified for a custom scroll bar button, an exception was thrown when the mouse pointer passed over the button.
4. [Fixed] When iGrid.NET was shown in a dialog that had a Cancel button that closes the form by the Escape key, the dialog was closed while the user was editing a text cell and pressed the Escape key to cancel the changes in the cell.

v1.60, build 0001 (the release of v1.6)

July 19th, 2006

1. [New] Now you can print and print-preview iGrid.NET by using a new component named **iGPrintManager**:



The following code shows how you can use this component to print-preview a grid in your application:

```
iGPrintManager myManager = new iGPrintManager();
myManager.PrintPreview(iGrid1);
```

This component is available for an extra price for all the iGrid.NET 1.60 users.

2. [New, Fixed] With this release we claim that our component works properly on 64-bit platforms under native 64-bit .NET Framework. We tested iGrid.NET 1.6 on the x64 edition of MS Windows for this purpose, and fixed all the bugs we found (it was just one bug, you can face it in the previous versions of our control if a tool tip was attached to a scroll bar custom button).
3. [New] **GetSpanColsOnLeft**, **GetSpanColsOnRight**, **GetSpanRowsOnTop**, and **GetSpanRowsOnBottom** methods were added to the **iGHeader** class. These methods allow you to determine how many columns and header rows are covered with a header cell on the each side of a column or header row.
4. [New] **DrawCellContents**, **DrawColHdrContents**, and **DrawGroupBox** methods were added to iGrid.NET. These methods allow you to draw the contents of a particular cell, column header, and group box on a specified graphics surface.
5. [New] **EffectiveBackColor**, **EffectiveForeColor** and **EffectiveFont** properties were added to the cell. These properties allow you to determine the real font and colors of a cell used when drawing.
6. [New] A **Height** property was added to the **iGHeader** class.
7. [Change, New] The **Redraw** property was marked as obsolete. The **BeginUpdate** and **EndUpdate** methods were added, use these methods instead of the **Redraw** property.
8. [Fixed] When an image and text were displayed in a cell and the text was over, above, or under the image, the automatic calculation of the cell width was incorrect. Fixed.

v1.50, build 0025

May 29th, 2006

1. [Fixed] A critical bug related to using of iGrid.NET in modal dialogs was fixed. An unexpected exception occurred if you used the previous builds of iGrid.NET in a modal dialog.
2. [Fixed] The iGrid.NET behavior in different screen DPIs was improved. In the previous builds when the auto-scale operation was performed on a form with iGrid.NET (it occurred if the form was designed in one DPI, for example with small fonts - 96DPI, and then run in another DPI, for example with large fonts - 120DPI), the scroll bars and XP-styled check boxes were incorrectly scaled and drawn.
3. [Fixed] The Group Row Level Styles dialog used for editing the **GroupRowLevelStyles** property in the property grid was improved. Now it is always shown in the center of screen and works properly at run-time.

4. [Change] The images in the iGrid.NET drop-down list are vertically centered now.
5. [New] The cell, column header, column and row properties are now categorized in the property grid.

v1.50, build 0021

March 22nd, 2006

1. [Fixed] When the layout of a non-moveable column was being restored through the **LayoutObject** object property, an exception was raised.
2. [Fixed] In the right-to-left mode the horizontal scroll bar was drawn incorrectly in the locked state (when its **Visibility** property was set to the **Always** value, and there were no horizontally scrollable contents in the grid).
3. [Fixed] In the right-to-left mode the Left and Right keys were mixed up.

v1.50, build 0018

March 3rd, 2006

1. [New] Icons for the **GoPrevPage** and **GoNextPage** actions were added. Now if you assign these actions to scroll bar custom buttons, iGrid.NET will draw their standard representations on the buttons.
2. [Fixed] Drop-down controls were shown incorrectly on multi-screen systems; they were always shown on the main screen.
3. [Fixed] A design-time bug was fixed. When a style was attached to a cell at design-time, an exception happened.

v1.50, build 0016

February 6th, 2006

1. [New, Change] The design-time support of iGrid.NET was improved so that it is now possible to use iGrid.NET in Visual C++ projects. The **iGColPattern**, **iGRowPattern**, **iGCellPattern**, **iGCellStyle**, and **iGColHdrStyle** classes are no longer serialized using their constructors with all the properties passed as parameters to them; now they are serialized using a constructor with one parameter or without a parameter, and their property values are assigned in several separate statements. The previous approach isn't supported by MS VC++ Windows Forms Designer (unknown and strange errors are generated during design time). The constructors mentioned above and the **SetColCellStyle** and **SetColColHdrStyle** methods of the column collection are marked as obsolete and will be removed in a new version of iGrid.NET. As for your existing code based on the previous builds of iGrid.NET, it will work properly with the new design-time approach.
2. [New] A new constructor was added to the **iGCellStyle** and **iGColHdrStyle** classes. This constructor has only one parameter that determines whether to set all the properties of the new object to “not set” (transparent) values or to default values.

v1.50, build 0014

December 31st, 2005

1. [Fixed] When an **iGCellStyleDesign** or **iGColHdrDesign** object had been attached to a column at design-time, and all the properties of the object had been set to the default values, code that attaches the style to the column was not generated.
2. [Fixed] When you clicked a combo or ellipsis button while editing a cell as text when the **SilentValidation** property was True, the combo or ellipsis button might not work.

v1.50, build 0012

December 15th, 2005

1. [New] Now iGrid.NET is fully supported in Visual Studio 2005 and can be used in the Windows Forms Designer without any limitations.
2. [New] The old versions of the **iGColPattern**, **iGCellPattern**, **iGCellStyle**, **iGColHdrStyle** constructors were re-added. This was done to simplify upgrade process from iGrid.NET 1.0.x (iGrid's columns created in the designer persist when you upgrade to iGrid.NET 1.5). These constructors are marked with the "Obsolete" attribute and will be removed in the next major update of the control.
3. [Improvement] The text-to-value converting algorithm was improved. If you entered a value that can be saved as a number in a text box cell, iGrid.NET might process this value enough long in debugging mode (up to 1 second).

4. [Fixed] When you set the **Selected** property of a cell to True, a wrong cell (the cell in the next column) was selected.
5. [Fixed] A Win32 exception was raised in the Windows 98/2000 environment if a combo button had been clicked.
6. [Fixed] The **Text** argument of the **RequestEdit** event worked incorrectly: if editing was started with a character key, the **Text** argument value was ignored.
7. [Fixed] Shadow under a drop-down window was shown independently of the system settings (the Show shadows under menus setting in the Display Properties dialog).

v1.50, build 0007 (the release of v1.5)

November 24th, 2005

1. [New] Now iGrid fully supports combo box cells. You can attach to its combo box cells standard drop-down lists which consist of items with text and/or an image, or even create your own drop-down lists from scratch:

A new **iGDropDownList** class is the built-in implementation of the drop-down list functionality. It allows you to create standard drop-down lists as well as lookup ones. Its **FillWithData** method allows you to populate it with the data from a DataTable, DataView, IDataReader, or IDbConnection object.

The following example shows how to create a lookup drop-down list, fill it from a data connection, and assign to a column:

```
…
//Create a drop down list
iGDropDownList myList = new iGDropDownList();
//Fill the drop down list from a data command
myList.FillWithData(sqlCommand1, "id", "name");

//Set the column's type to combo
myCol.CellStyle.Type = iGCellType.Combo;
//Assign the drop down list to the column
myCol.CellStyle.DropDownControl = myList;
…
```

In fact, the **iGDropDownList** class implements the new **IiGDropDownControl** interface you can also use to create your own drop-down lists and assign them to the iGrid combo box cells through the new **DropDownControl** property of an **iGCellStyle** object or a cell.

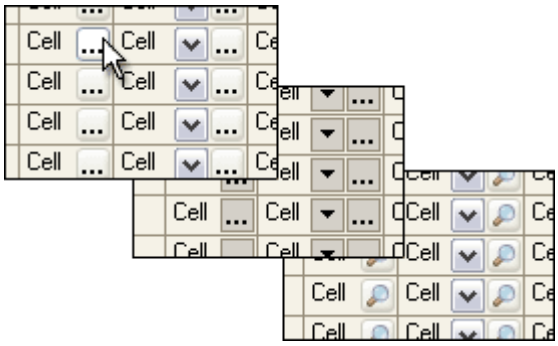
2. [New] The **iGTypeFlags** enumeration was extended with a **ComboEditable** field which allows you to specify whether a combo box cell can be edited as text. The following example shows how to make such a column:

```
…
//Create a column
iGCol myCol = iGrid1.Cols.Add();
//Set the column type to combo
myCol.CellStyle.Type = iGCellType.Combo;
//Make the column's cells editable as text
myCol.CellStyle.TypeFlags = iGCellTypeFlags.ComboEditable;
//Assing a drop down control to the column
myCol.CellStyle.DropDownControl = myList;
…
```

3. [New] For combo box cells, iGrid stores the reference to the selected drop-down list item in the new **AuxValue** property of a cell. It has the object data type and can be used freely for other types of cells. For instance, you can use it to store an additional value for a cell (like with a Tag property).

   A new **ByAuxValue** field was added to the **iGSortType** enumeration to give you the ability to sort and group the grid by the **AuxValue** property values.

4. [New] Now an ellipsis button can be displayed in a cell of any type:



   To show the ellipsis button in a cell, specify a new **HasEllipsisBtn** flag in the **TypeFlags** property of the cell or style attached to the cell. The following example demonstrates how to show ellipsis buttons in the cells of the first column:

```
iGrid1.Cols[0].CellStyle.TypeFlags = iGCellTypeFlags.HasEllipsisBtn;
```

   A new **EllipsisBtnClick** event is intended to handle the clicks on the ellipsis buttons.

   By default the ellipsis buttons display an ellipsis in them, but you can change this in one of the following ways. First, you can specify your own image for the ellipsis buttons using the **EllipsisBtnGlyph** property of iGrid. The following example attaches an image to the ellipsis buttons:

```
iGrid1.EllipsisBtnGlyph = myImage;
```

   Second, new **DrawEllipsisBtnForeground** and **DrawEllipsisBtnBackground** events allow you to make your own drawing for ellipsis buttons.

5. [New] A new **LayoutObject** property allows you to save and restore the columns' order, visibility, width, sorting, and grouping. This property returns the **iGLayout** object. Using this object you can save and restore the layout to a file or any other source. The following example shows how to save and restore the layout to a file:

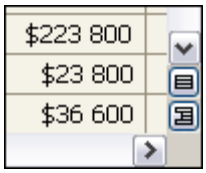
```
private void Form1_Load(object sender, System.EventArgs e)
{
  …
  StreamWriter myWriter = new StreamWriter("iGrid1Layout.xml", false);
  myWriter.Write(iGrid1.LayoutObject.Text);
  myWriter.Close();
  …
}
private void Form1_Closed(object sender, System.EventArgs e)
{
  …
  if(File.Exists("iGrid1Layout.xml"))
  {
    StreamReader myReader = new StreamReader("iGrid1Layout.xml");
    iGrid1.LayoutObject.Text = myReader.ReadToEnd();
    myReader.Close();
  }
  …
```

6. [New] New **SelectAll**, **UnselectAll**, **ExpandAll** and **CollapseAll** fields were added to the **iGAction** enumeration. Now you can select/unselect all the cells and expand/collapse all the rows in one code line. The following example shows how to select all the cells in an iGrid control:

```
iGrid1.PerformAction(iGActions.SelectAll);
```

7. [New] A new **Action** property of **iGScrollBarCustBtn** was implemented. After you assign an action to a scroll bar custom button with this property, the button will automatically execute this action and will draw its graphic representation. iGrid supports graphical representations for the following actions: go first/last column/row; expand/collapse all; select/unselect all cells.

   The following picture illustrates how the "expand/collapse all" actions are drawn:



   The following code example shows how to create such buttons:

```
iGrid1.VScrollBar.CustomButtons.Add(
  iGScrollBarCustBtnAlign.Far,
  iGActions.ExpandAll,
  "Expand all");
iGrid1.VScrollBar.CustomButtons.Add(
  iGScrollBarCustBtnAlign.Far,
  iGActions.CollapseAll,
  "Collapse all");
```

8. [New] Now you can display icons of different sizes in the cells of the same grid. A new **ImageList** property of the cell class and cell style class allows you to do this:

   The following example shows how to assign different image lists to the first and second columns:

```
iGrid1.Cols[0].CellStyle.ImageList = imageList1;
iGrid1.Cols[1].CellStyle.ImageList = imageList2;
```

9. [New] A new **ValueType** property which simplifies editing of the iGrid cells was added to the **iGCellStyle** class. After you specify this property, iGrid will automatically convert entered text to the specified type, regardless of the previous cell value (this may be helpful when the old cell value was DBNull or was not assigned). The following example shows how to create a column that accepts from the user values only of the double data type:

```
iGrid1.Cols[0].CellStyle.ValueType = typeof(double);
```

10. [New] A new **GetPreferredRowHeight** method of iGrid allows you to obtain the minimal required row height taking into account the column styles before iGrid is populated. It is useful if you need to have

rows of enough height to display your data if the standard row height isn't enough. The following example shows how to set the default row height before rows are added:

```
…
//add the columns
…
iGrid1.DefaultRow.Height = iGrid1.GetPreferredRowHeight(true, true);
…
//add the rows
…
```

11. [New] This version of iGrid introduces the single-click edit mode in addition to the double-click edit mode. To start edit a cell in the single click mode, the user should just click a cell (it is not important whether it is selected or not), but in the non-single click mode only the current cell will be edited after a click (otherwise it will be just made current but not edited). Use a new **SingleClickEdit** property of iGrid to specify this behavior for all cells or the same property of **iGCellStyle** to do it for particular cells.

    In the single-click edit mode you can prevent a cell from editing with the **DoDefault** parameter of the **CellMouseDown** event.

12. [New] A new **showDropDown** parameter of the **RequestEditCurCell** method allows you to specify how to edit a combo cell - with the attached drop-down control or as text. The following example demonstrates how to show a drop-down control for a combo box cell programmatically:

```
iGrid1.RequestEditCurCell(true);
```

13. [New] The **iGCell** class has a new **Text** property. This property allows you to obtain the text displayed in a cell.

14. [New] A new **Text** field was added to the **RequestEdit** event's arguments. Using this field you can specify the text to be displayed in the text box when editing a cell. The following example demonstrates how to show the zero in the text box when a cell is edited and its value is null (Nothing in VB):

```
private void iGrid1_RequestEdit(object sender,
TenTec.Windows.iGridLib.iGRequestEditEventArgs e)
{
  if(iGrid1.Cells[e.RowIndex, e.ColIndex].Value == null)
  e.Text = "0";
}
```

15. [New] By default iGrid suppresses any unexpected internal errors and does not display the corresponding error message. A new **DebugMode** property of iGrid allows you to control this behavior. If you have noticed an abnormal behavior of iGrid, set the **DebugMode** property to True; in this case iGrid will display an error message instead of suppressing it, and you can send this description to our customer support service in order to help us fix the problem as soon as possible.

16. [New] A new **EmptyStringAs** property of the **iGEmptyStringAs** enum type was added to the **iGCellStyle** class. It allows you to specify how to interpret an empty string when it is entered into a cell – as DBNull, numeric zero value, Null or simply as the zero-length string.

17. [New,Change] A new **GridLines** object property of iGrid was implemented. The purpose of this property is to store all the settings of different grid lines you can see in iGrid. Some new grid line settings were implemented in this class, and the **VGridLines**, **HGridLines**, **ExtendGridLines**, and **GridLines** properties from the previous version were transferred into this new property.

    Now you can set different styles for the group rows' grid lines. To do this, use the **GroupRows** property of the **GridLines** object. The following example shows how to make group rows' grid lines thick:

```
iGrid1.GridLines.GroupRows.Width = 3;
```

    New **HorizontalExtended** and **VerticalExtended** properties of the **GridLines** object allow you to set the style of the vertical and horizontal extended grid lines. The following example makes the extended grid lines dim:

```
//Extend the grid lines
iGrid1.GridLines.ExtendMode = iGGridLinesMode.Both;
//Set the color of the extended grid lines
iGrid1.GridLines.HorizontalExtended.Color = SystemColors.Control;
iGrid1.GridLines.VerticalExtended.Color = SystemColors.Control;
```

    New **HorizontalLastRow** and **HorizontalLastCol** properties of the **GridLines** object allow you to set the style of the grid lines that frame the cells. The following example shows how to make the grid lines that frame the cells thick:

```
iGrid1.GridLines.VerticalLastCol.Width = 3;
iGrid1.GridLines.HorizontalLastRow.Width = 3;
```

18. [New] Now you can track a lot of events related to iGrid.NET's scrollbars. New **VScrollBarScroll**, **VScrollBarValueChanged**, **HScrollBarScroll** and **HScrollBarValueChanged** events allow you to track the movement of the thumb box on a scroll bar. New **VScrollBarVisibleChanged**, **HScrollBarVisibleChanged**, **VScrollBarWidthChanged**, and **HScrollBarHeightChanged** events allow you to track visibility and thickness changes in the scroll bars.
19. [New] A new **CurRowChanged** event was added. Using this event you can track the changes of the current row.
20. [New] The **CellClick**, **CellDoubleClick**, **CellMouseDown**, and **CellMouseUp** events now have a new **Control** argument that provides you with the information about the control under the mouse pointer. For example, if the mouse is moved over the ellipsis button, the **Control** argument of the **CellMouseMove** event will be equal to the **iGControl.EllipsisBtn** value; if the mouse is over the combo button, the value will be **iGControl.ComboBtn**.
21. [New,Change] A new **Kind** parameter was added to the **ColHdrMouseEnter, ColHdrMouseMove, ColHdrMouseDown, ColHdrMouseUp, ColHdrClick, ColHdrDoubleClick**, and **ColHdrMouseLeave** events. Using this parameter, you can determine whether the mouse pointer is over a header of a normal column, over the header of the row text column (it can be visible in the group box), or over the extra column header (it is displayed in the header area not occupied by the columns).

    The **colIndex** argument of the **ColHdrMouseEnter, ColHdrMouseMove, ColHdrMouseDown, ColHdrMouseUp, ColHdrClick, ColHdrDoubleClick**, and **ColHdrMouseLeave** events now returns -1 when the mouse pointer is over the header area not occupied by a column. The -2 value was returned in the previous versions indicating extra column header, but in this version you should use the new **Kind** parameter of those events instead.
22. [New,Change] The **Current** property of the cell was removed. Use new **SetCurCell** and **SetCurRow** methods instead. The following example shows how to set the current cell:

    ```
    iGrid1.SetCurCell(5, "Country");
    ```

    To change the current row, do the following:

    ```
    iGrid1.SetCurRow(5);
    ```
23. [Change] The **NoWrap** flag in the **iGStringFormatFlags** enumeration was replaced with **WordWrap**. In the previous build the **NoWrap** flag was set indicating that the cell text should be displayed on one line; as the equivalent for this setting in the new version the **WordWrap** flag isn't specified by default.
24. [Change] The **ShowControlsInAllCells** property equals True by default (in the previous versions the default value was False).
25. [Change] The constructors of the **iGColPattern**, **iGCellPattern**, **iGCellStyle**, and **iGColHdrStyle** classes were changed, and the code generated automatically by the Windows Forms designer for a previous version of the grid will not work.
26. [Change] The licensing procedure was changed. Now the license is not automatically embedded to the resource of an application, it must be specified programmatically in the constructor of the grid. This allows you to use the grid without any licensing problems not only in exe applications, but in any other types of applications including DLL.

    If you create iGrid with visual designer, the license string is automatically specified in its constructor.
27. [Change] The current version of iGrid no longer supports the constructor with the **pageCapacity** parameter.
28. [Change] The combo box button is now shown in the current cell when the grid is not focused and the **ShowControlsInAllCells** property equals False.
29. [Change] The **iGAutoWithColMode** enumeration was renamed to **iGAutoWidthColMode**.
30. [Fixed] In the **MultiSimple** selection mode the Enter key started editing.
31. [Fixed] When the columns in the frozen area were not included in selection, the focus rectangle in the row mode was drawn improperly.
32. [Fixed] iGrid was not destructed before the application was closed.
33. [Fixed] The **IndexOf** method of the **iGSelectedCellsCollection** class could return an incorrect value.
34. [Fixed] Now when you press one of the arrow or HOME/END keys, the cells in a zero width column are skipped and the current cell selection moves to the next visible cell.
35. [Fixed] The focus rectangle in the row mode was not drawn for the manual group rows.

36. [Fixed] When the current cell is changed programmatically and the cell is being edited, iGrid cancels the editing. In the previous builds, the editing was not finished.
37. [Fixed] When a column is moved programmatically and a cell is being edited, iGrid cancels editing the cell. In the previous builds, the editing was not finished.
38. [Fixed] After iGrid was sorted, the current cell might be invisible if there were frozen rows in iGrid.
39. [Fixed] iGrid could raise an exception when you removed a column with a selected cell while the grid had been grouped.
40. [Fixed] In the stay-sorted mode, after a cell's value or image index was changed, the **RowIndex** argument of the **AfterCommitEdit** event had an improper value if the row was moved.
41. [Fixed] In the old versions of iGrid.NET, when the user pressed the PAGEUP or PAGEDOWN keys, the grid was scrolled for a little larger distance that it was needed.

v1.00, build 0007

July 4th, 2005

1. [Fixed] The previous build did not allow you to prohibit the default key processing in the **KeyDown** event.
2. [Fixed] You could not change the new value stored in the currently edited cell through the **NewValue** field in the **BeforeCommitEdit** event.
3. [Fixed] When you added new rows to the grid in the **BeforeRowStateChanged** event when the row was expanded/collapsed, the plus/minus sign button in the row wasn't updated.
4. [Fixed] iGrid might not change its view when the system appearance changed.
5. [Fixed] Bug when trying to start editing of a text cell with invisible text.
6. [Change] The designer verb "Edit Data..." was separated into the "Cols..." and "Rows & Cells…" verbs. The corresponding editors are invoked immediately without the intermediate dialog now.
7. [Fixed] If the last column had zero width, the user could not resize it.
8. [Fixed] When a scroll bar custom button is not enabled, the mouse events such as **MouseMove** are not raised for it (the previous versions raised them).
9. [Fixed] When you accessed the **SelectedCells** property in code and clicked a column header while the grid had no selected cells, an exception was raised.
10. [Fixed] When the left mouse button was pressed over a cell, the **CellMouseMove** event had incorrect mouse pointer coordinates.
11. [Fixed] When a row was moved with the **Move** method, the row keys might be incorrect.
12. [Fixed] When the mouse wheel is rotated, iGrid is scrolled by the small change of the vertical scroll bar. Before there was a bug and the grid was scrolled by the small change of the horizontal scroll bar.
13. [Fixed] When the **RequestEditCurCell** method is invoked, the **RequestEdit** event is raised (in the previous version it was not raised).

v1.00, build 0000 (the release of v1.0)

April 20th, 2005

The first official version released.