10Tec iGrid.NET 11.1
What's New in Control

Contents

v11.1.0 | 2023-Nov-07 (Release)

Most Important Improvements Briefly

- Opening of the AutoFilterManager filter box for huge grids no longer makes the UI non-responsive.
- Great performance improvements for the filter box search functionality (usable for 500'000+ rows).
- Ability to detect printing in custom drawing and dynamic formatting events for special print effects.
- Smarter Tab key behavior in drop-down lists that saves time and keystrokes while editing huge grids.
- New simple PrintManager setting for centering iGrid on paper.
- The search-as-type tools now better support languages with digraphs (Hungarian, German, etc.)

The Minimal Supported Version of .NET Framework

Starting from this release of iGrid, the minimal supported version of .NET Framework is 4.5.

Common Search-as-Type Improvements

This release greatly improves the built-in search-as-type functionality used in the core grid control and its AutoFilterManager add-on. The improvements are mainly aimed at better support for languages with specific rules concerning combinations of two or more letters. Among them are digraphs representing one character in the Hungarian language (such as 'sz' and 'dz') and double 'ss' that may be used instead of 'ß' in German. These improvements work in the following tools:

1) The search-as-type functionality for normal iGrid cells.
2) The search-as-type functionality for items of built-in drop-down lists.
3) The cell auto-complete feature that can be enabled for text editing.
4) The search-as-type functionality for the pick list in the AutoFilterManager filter box.
5) The search box above the pick list in the AutoFilterManager filter box.

The details of the improvements are given below.

1. [Change] The previous versions of iGrid allowed you to enter only one character that results in no matches during a search-as-type operation. Such a character was displayed in red in the search window, and the next entered symbol replaced this red symbol. This did not allow you to enter two or more characters that combined together could result in a match. This restriction has been removed in this release and now you can enter as many characters as you need for a culture-specific search. The whole search text is also displayed in red because in the general case its ending part that leads to missing search results may vary with every entered character due to the above-mentioned language peculiarities.

2. [New] Now all search-as-type tools listed above use the culture-specific rules of string comparison to provide correct search results for the current culture. However, in some cases the user may prefer to have traditional letter-by-letter search. For example, the letters 's' and 'z' forming a digraph in the Hungarian language may be treated as independent letters even if they are written one after another.

   iGrid can be configured to provide this functionality with the new **SearchAsTypeStringComparison** property of the .NET System.StringComparison type. Its default value is **StringComparison.CurrentCultureIgnoreCase**, which defines the default search-as-type behavior. To disable the string comparison rules used in the current culture and provide independent processing of letters, you can specify the rules of the so-called invariant culture like this:

   ```
   iGrid1.SearchAsTypeStringComparison =
     StringComparison.InvariantCultureIgnoreCase;
   ```

   Note that the **StringComparison** enumeration contains such members as **CurrentCulture** and **InvariantCulture** that can be used to provide case-sensitive search-as-type functionality in iGrid.

   Pay also attention to the fact that the **SearchAsTypeStringComparison** property affects all 5 search-as-type tools listed in the beginning of this section.

3. [Enhancement][Change] Assigning of the same string to the **iGrid.SearchAsType.SearchText** property did nothing in the previous versions. Now the search-as-type operation is repeated in this case. This allows you to repeat search in the grid with updated data if the search criteria didn't change.

4. [Enhancement] Now you can specify invisible columns in the **iGrid.SearchAsType.SearchCol** property and filter grids by values from such columns from code with the **iGrid.SearchAsType** object.

5. [Fixed] If you assigned a string to the **iGrid.SearchAsType.SearchText** property to activate search-as-type and this string resulted in no matches, the search text in the search window was rendered in black as if there were matches. Now the whole search text that results in no matches is displayed in red.

6. [Fixed] The keyboard hint in the search-as-type window was not drawn correctly in the Right-to-Left mode.

iGrid Improvements

1. [New] iGrid implements the **ColHdrDynamicFormatting** event that can be used to specify column header formatting dynamically.

2. [New] The event arguments objects of the main iGrid events related to drawing implement the new Boolean **IsPrinting** property indicating whether these events are fired while iGrid is being printed on paper. This allows you to implement specific color and font effects during printing. For example, the following event handler can be used to print grid cells in monochrome mode regardless of the colors used to render them on the screen:

```
private void iGrid1_CellDynamicFormatting(
  object sender, iGCellDynamicFormattingEventArgs e)
{
  if (e.IsPrinting)
  {
    e.ForeColor = Color.Black;
    e.BackColor = Color.White;
  }
}
```

The events that provide the new functionality are listed below.

- The dynamic formatting events: **CellDynamicFormatting**, **ColHdrDynamicFormatting**, **FooterCellDynamicFormatting**.
- The custom draw events used to draw 3 types of iGrid cells: **CustomDrawCellBackground**, **CustomDrawCellForeground**, **CustomDrawColHdrBackground**, **CustomDrawColHdrForeground**, **CustomDrawFooterCellBackground**, **CustomDrawFooterCellForeground**.
- The combo button custom draw events: **CustomDrawCellComboButton**, **CustomDrawColHdComboButton**.
- The ellipsis button custom draw events: **CustomDrawCellEllipsisButtonBackground**, **CustomDrawCellEllipsisButtonForeground**.
- The event for drawing custom contents in the footer's row header: **CustomDrawFooterRowHdr**.

3. [New] The event arguments of the **CustomDrawFooterCellBackground** and **CustomDrawFooterCellForeground** events implement the new **State** property of the **iGControlState** type. The value of this property can be tested in event handlers of these events to determine whether the drawn footer cell has the enabled or disabled state.

4. [New] The **iGHdrRow** class representing a row in the iGrid header now implements the **Cells** property. It returns the enumerable collection of column headers in a particular header row (objects of the **iGColHdr** class). This feature can be used to write neat code that initializes column headers in a multi-row header like in the following sample:

```
var hr = iGrid1.Header.Rows.Add();
hr.Cells[0].Value = "Column title 1";
hr.Cells[1].Value = "Column title 2";
```

In the previous versions you would need to use the **iGrid.Header.Cells[]** collection indexed by row and column indices to do the same:

```
iGrid1.Header.Rows.Add();
int myNewHdrRowIdx = iGrid1.Header.Rows.Count - 1;
iGrid1.Header.Cells[myNewHdrRowIdx, 0].Value = "Column title 1";
iGrid1.Header.Cells[myNewHdrRowIdx, 1].Value = "Column title 2";
```

5. [New] The new **DrawFooterRowHdrContents()** method was added to iGrid:

```
public void DrawFooterRowHdrContents(
  Graphics g,
  int x, int y, int width, int height,
  bool adjustForPrinter)
```

It is actually a wrapper for the **CustomDrawFooterRowHdr** event of iGrid the developer can use to draw custom contents in the row header area inside the footer section. This method was implemented as a part of the iGrid printing infrastructure to provide the ability to draw custom contents in the footer row header from the PrintManager add-on.

6. [Enhancement][Change] In the previous versions of iGrid pressing the Tab key in iGrid drop-down lists led to selection of the next item in the list. Now Tab selects the highlighted item in the edited cell, finishes editing and moves the current cell selection to the next available cell. Shift+Tab does almost the same except moving the current cell selection to the previously available cell.

   The new behavior of the Tab key fully duplicates the corresponding behavior of the Tab key in Windows combo box controls. This feature also minimizes the number of keystrokes used to edit grid cells, which can lead to big savings in time and keystrokes if your users enter big amounts of data in forms based on grids with combo box cells.

7. [Enhancement][Change] The default value of the **AddTreeButtons** property of the **iGFillWithDataOptions** class used to specify options for the **FillWithData()** method of iGrid was changed from False to True. This makes sense and helps to write less code because as a rule **AddTreeButtons** is set to True if a value is specified for the **RowLevelCol** property of an **iGFillWithDataOptions** object. This also corresponds to the behavior of the **FillWithData()** method in iGrid of the version 10.0 or earlier and helps keep the code backward compatible.

8. [Change][Code-Upgrade] To support the functionality of the new **iGPrintingOptions.SystemElemControls** flag introduced in the PrintManager add-on in this release (see a more detailed description in the section dedicated to the PrintManager enhancements), the new Boolean **plainElemControls** parameter was added to the **DrawGroupBox** and **DrawColHdrContents** methods.

   The **plainCheckBox** parameter of the **DrawCellContents** method was renamed to **plainElemControls** and now it specifies whether all cell controls (check box, combo and ellipsis buttons) are rendered using the plain look. If you do not use the 'plainCheckBox' parameter name explicitly in your code (in named arguments, etc.), no changes in source code are required.

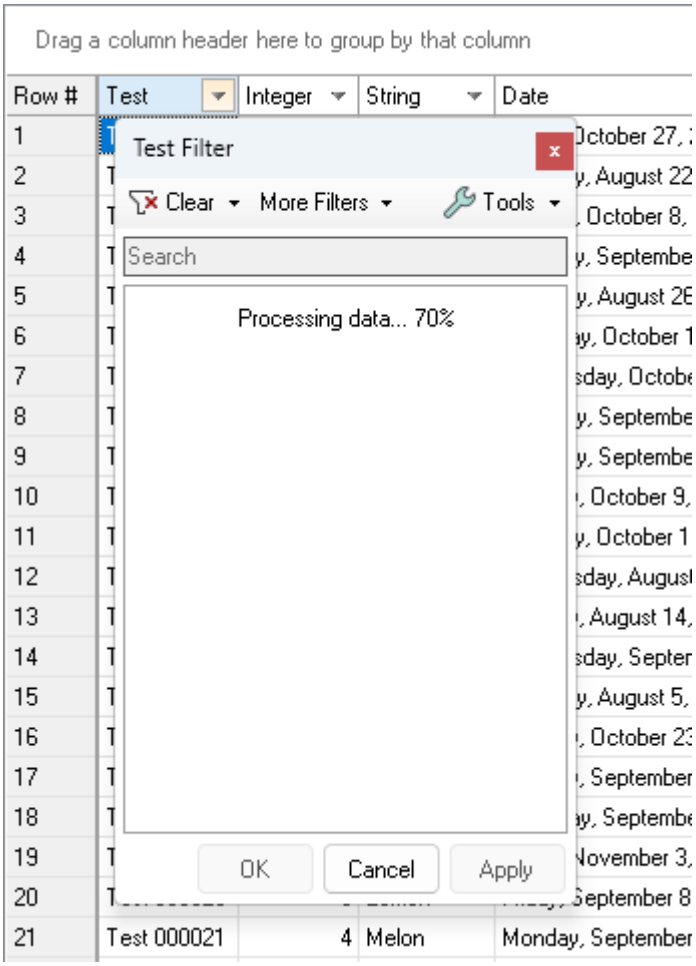
9. [Fixed] The **ColIndex** property of the event arguments of the **IncludeRowInTotalsCalculation** event contained an incorrect value.

10. [Fixed] The **AddRange()** and **InsertRange()** methods of the **iGrid.Cols** column collection object returned incorrect values.

AutoFilterManager Improvements

1. [Enhancement][New] The filter box functionality was significantly changed to make the UI responsive when AutoFilterManager is attached to huge grids containing hundreds of thousands of rows and/or when it works on budget computers with slow performance. In the previous versions of the add-on the user may have seen significant delays between the moments when they clicked the filter box button and the filter box appeared on the screen. This happened because AutoFilterManager needs to scan all grid rows to build the filter item list for a column. This complex process includes building the list of unique items, sorting it alphabetically, taking into account the visibility of grid rows that may be already filtered out by filters in other columns, etc. All this may take a significant amount of time and lead to delays in the UI.

This release of AutoFilterManager builds the filter item list in a background thread not to block the UI. Autofiltermanager waits half a second for this job to complete. If it was not enough, the filter box is opened anyway and becomes available for user actions. The list building process is finishing in the background, and the corresponding message informing about the progress of the process every 5% appears inside the filter item list:



This maximum display delay of the filter box can be adjusted with the new **FilterBoxMaxDisplayDelay** property of the **iGAutoFilterManager** class. It is an integer property that specifies a value in milliseconds. The default value is set to 500 milliseconds.

The template of the message displaying the list building progress can be changed/localized with the new string property **iGAutoFilterManager.UIStrings.ItemList.BuildingProgressTemplate**. Its default value is set to "Processing data… {0}%". The "{0}" substring in this string is replaced with the percentage of completion when displayed on the screen.

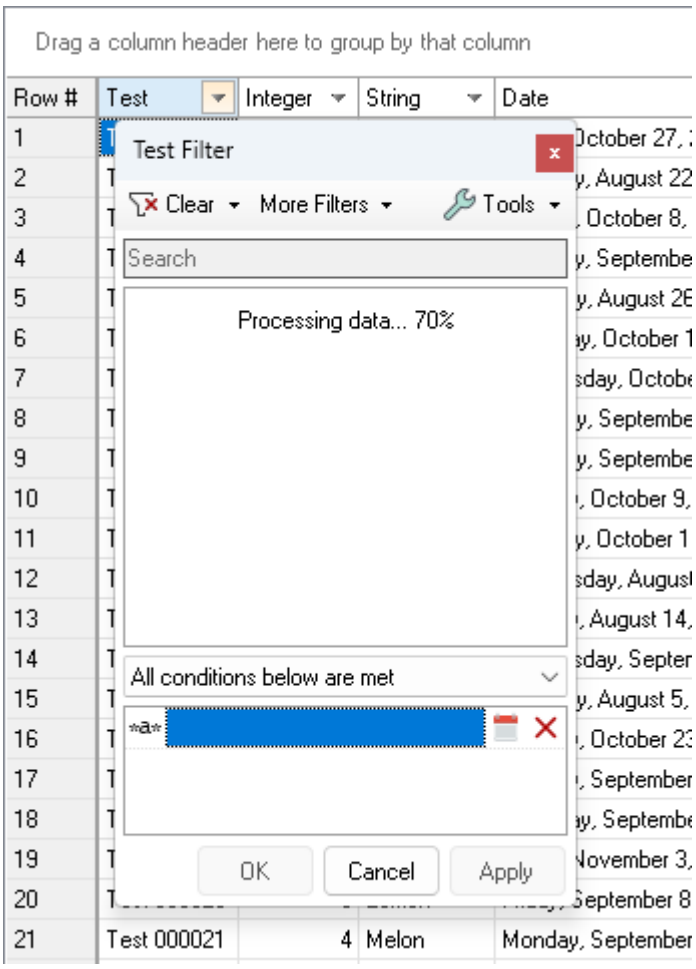
2. [Optimization] The search box performance has been greatly optimized: now filtering with it works up to 25 times faster. This makes the search functionality usable for lists containing 100'000+ items.

3. [New] The **iGAutoFilterManager** class implements the new **CustomConditionSuggestion** property of the **Nullable<iGFilterConditionOperator>** type. If this property is set to a value from the **iGFilterConditionOperator** enumeration, AutoFilterManager automatically creates a new custom filter condition with the specified operator in the filter box's list of custom filter conditions if this list was empty. The default value of this property is null (Nothing in VB), which means custom filter conditions are not created automatically the way described above.

   Setting the **CustomConditionSuggestion** property to a non-null value may help your users to start using the filter box immediately after opening while the grid data are being processed to build the filter item list. For example, the following setting

```
iGAutoFilterManager1.CustomConditionSuggestion =
  iGFilterConditionOperator.Contains;
```

will result in the following state of the filter box while its filter item list is being populated:



Note that the user can already enter a value for the parameter of the Contains operator and click the OK or Apply button to apply the filter – even if the filter item list is not populated by that time yet.

4. [New] The **iGAutoFilterManager** class implements the new **FilterBoxDynamicCaption** event that can be used to specify the caption of the filter box dynamically. The corresponding event arguments class provides 2 properties:

```
public class iGFilterBoxDynamicCaptionEventArgs : EventArgs
{
  public readonly int ColIndex;
  public string Caption;
}
```

The **ColIndex** property is the index of the column the filter box is displayed for. The **Caption** property can be used to change the default filter box caption built based on the template stored in the **iGAutoFilterManager.UIStrings.MainControlsTexts.FilterBoxCaptionTemplate** property.

5. [Enhancement] The availability of the OK and Apply buttons in the filter box is now evaluated during text editing of custom filter condition parameters. The algorithm checks whether the parameter of a condition is empty, and if so, the condition is skipped.

   The availability of the OK and Apply buttons is also synched now because they actually perform the same action with the only difference that the Apply button does not close the filter box.

6. [Enhancement] The combo box above the custom condition list specifying how custom filter conditions are combined now contain more elegant formulations:
   - All conditions below are met
   - At least one condition below is met
7. [Enhancement] If you activated incremental search in the filter item list by typing some first characters of the item you wanted to find, pressing Escape closed the filter box in the previous versions of AutoFilterManager. Now this key does not close the filter box while incremental search is active – it clears the search text instead of closing the dialog.

   The filter item list allows you to check/uncheck items with the Space key. However, this functionality did not allow you to enter a space character during incremental search in the previous versions of AutoFilterManager. In this release the Space key adds a space character to the search string during incremental search, and the highlighted item can be checked/unchecked using the Shift+Space combination.
8. [Enhancement] This release of AutoFilterManager allows you to clear the contents of the search box above the filter item list if it is focused and not empty when you press the Escape key. In the previous versions the filter box dialog closed in this case.
9. [Change] The check mark for the first special "(Select All)" item in the filter item list is not cleared automatically when you add custom conditions. It is done only after pressing the OK button and if there are defined custom filter conditions.
10. [Fixed] AutoFilterManager added its tooltips displaying column filters to x-buttons of column headers placed in iGrid's group box.
11. [Fixed] AutoFilterManager crashed if a column header did not have the filter box button and this column header was dragged to the group box.
12. [Fixed] The check state of the first special "(Select All)" item in the filter item list was not updated properly when checking/unchecking items if the list was filtered using the search box above it.
13. [Fixed] At design time, if an iGrid was selected in the **Grid** property of an **iGAutoFilterManager** component and you removed that grid from the form, the corresponding **Grid** property in the property grid was not cleared correctly. Moreover, you could not select another iGrid in the same **Grid** property - an error message with the text "Property value is not valid" appeared.

PrintManager Improvements

1. [New] This release of PrintManager allows you to specify the horizontal alignment for the grid on paper. It is done with the new **GridAlignment** property of the **iGPrintManager** class. This property accepts one of the following values from the new **iGPrintGridAlignment** enumeration:

```
public enum iGPrintGridAlignment
{
  Near,
  Center,
  Far
}
```

   The position of the grid on paper specified with the **Near** and **Far** options depend on the Right-To-Left status of the grid. If the grid is not Right-To-Left, **Near** aligns it to the left edge of the printable area, and **Far** aligns it to the right edge. If the grid is Right-To-Left, **Near** aligns it to the right edge, and **Far** aligns it to the left edge.

   The default value of the **iGPrintManager.GridAlignment** property is **iGPrintGridAlignment.Near**, which corresponds the behavior of PrintManager in the previous versions.

Note that the horizontal alignment is done in the printable area of the page but not the whole page. This means that if you want to get the correct position of the grid in the center of the page, the left and right non-printable margins of the page must be equal.

2. [Change][Code-Upgrade] In the previous versions of PrintManager it was not possible to specify whether cell combo and ellipsis buttons should be printed using the system style or plain look as it was done for cell check boxes with the **SystemCheckBox** flag from the **iGPrintingOptions** enumeration. The functionality of this flag was extended in this release, and now it specifies the look of combo and ellipsis buttons too because all cell controls should be rendered using the same visual theme. The flag was renamed to **SystemElemControls** to indicate its purpose. To upgrade existing code, simply replace all occurrences of the **iGPrintingOptions.SystemCheckBox** flag with **iGPrintingOptions.SystemElemControls.**

3. [Fixed][Change] In the previous versions of PrintManager it was possible to launch an infinite loop of generating empty pages after attaching the following event handler to the **PrintPage** event of the **iGPrintManager.Document** object representing the print job:

```
private void Document_PrintPage(object sender, PrintPageEventArgs e)
{
  e.HasMorePages = true;
}
```

This problem has been fixed in this release after rewriting the corresponding part of the internal infrastructure. One of the related changes is that now **iGPrintManager.Document** returns an object of the new **iGPrintDocument** type instead of the standard .NET **PrintDocument** class. The **iGPrintDocument** class is a direct ancestor of the **PrintDocument** class, so that all the features are preserved and no changes in existing code are required.

4. [Enhancement] The **Grid** property of the **iGPrintManager** component is now displayed in the dedicated category titled "Grid" in the property browser – similarly to the same property of the **iGAutoFilterManager** component.

5. [Fixed] Custom contents of the footer row header rendered in an event handler of the **CustomDrawFooterRowHdr** event of iGrid were not printed.

6. [Fixed] If the **iGPrintingOptions.DrawSystemStyledHeader** was not specified in the **PrintingOptions** property of an **iGPrintManager** object (which is also the case if the plain printing style is used), column headers inside the group box had incorrect height on paper.

7. [Fixed] The **iGFitColsOnPage.ScaleGrid** setting specified in the **iGPrintManager.FitColsOnPage** property did not work correctly for right-to-left grids.

8. [Fixed] At design time, if an iGrid was selected in the **Grid** property of an **iGPrintManager** component and you removed that grid from the form, the corresponding **Grid** property in the property grid was not cleared correctly.

Addendum: Tags used to classify changes

[New]	-	Completely new feature.
[Change]	-	Change in a member functionality or interactive behavior.
[Fixed]	-	Fixed bug or solved problem.
[Removed]	-	Member was completely removed.
[Enhancement]	-	Some functionality was enhanced.
[Optimization]	-	Performance was improved.

[Renaming] - Member was renamed.

[Code-Upgrade] - Source code for previous versions may require changes.