



TDBADVGRID DEVELOPERS GUIDE

Documentation : Oct, 2010
Copyright © 1996 - 2010 by tmssoftware.com bvba
Web: <http://www.tmssoftware.com>
Email : info@tmssoftware.com

Table of contents

TDBAdvGrid availability.....	3
TDBAdvGrid use	3
TDBAdvGrid architecture	4
TDBAdvGrid setup	5
TDBAdvGrid sorting.....	7
TDBAdvGrid HTML Templates.....	8
TDBAdvGrid memo fields, Boolean fields & image blobs	9
TDBAdvGrid editing.....	10
TDBAdvGrid Unicode support	11

TDBAdvGrid availability

TDBAdvGrid is available as VCL and VCL.NET component.

VCL versions:

TDBAdvGrid is available for Delphi 5,6,7,2005,2006,2007,2009,2010,XE and C++Builder 5,6,2006,2007,2009,2010,XE..

TDBAdvGrid has been designed for and tested with: Windows 2000, 2003, 2008, XP, Vista, Windows 7.

VCL.NET versions:

TDBAdvGrid is available for Delphi 2005, Delphi 2006, Delphi 2007.

TDBAdvGrid use

The TMS TDBAdvGrid component is designed to be used in the most broad types of applications needing to display or handle data in rows and columns. TDBAdvGrid descends from TAdvStringGrid and as such inherits a lot of functionality of TAdvStringGrid. Please refer to the TAdvStringGrid developers guide, help file and samples for information specifically about the base grid features.

TDBAdvGrid architecture

TDBAdvGrid descends from TAdvStringGrid. TAdvStringGrid is a grid designed to show & edit data that is either stored in cells or virtual cells. The number of cells displayed (rows x columns) in the grid always reflects the number of cells or virtual cells in the grid. Through a Columns collection, property `grid.Columns[ColumnIndex].FieldName`, it can be configured what DB fields to display in each column. With this Columns property it is possible to define things like color, font, editor type, formatting, etc... for a column in the grid. TDBAdvGrid adds data-awareness to TAdvStringGrid in two distinct ways:

TDBAdvGrid with PageMode property set true

This is the default mode for TDBAdvGrid. With `PageMode = true`, the behaviour of TDBAdvGrid most closely resembles the behaviour of the Borland TDBGrid. This means that TDBAdvGrid only loads the rows from the connected dataset that are visible. The grid will load a new page of rows when it is scrolled to a previous or next page. As the grid loads the minimum number of rows that is simultaneously required for display, this is the fastest mode. A disadvantage of this mode is that not all rows are simultaneously available for operations like searching, grouping, filtering, sorting, export, printing.

TDBAdvGrid with PageMode property set false

When `PageMode` is false, all rows of the dataset are loaded in the grid when the dataset is activated. This can have an initial performance hit for large datasets. After all data is loaded from the dataset, the grid is disconnected from the dataset. The full base class TAdvStringGrid operations on the data can now be directly executed. A disadvantage is that the grid will not operate synchronously with the dataset cursor and cannot edit the dataset.

It will depend on the requirements of your applications what mode is recommended. For typical reporting scenarios where user just needs to view, sort, print, export data the grid with `PageMode` set to false is adequate and operates with minimum amount of code to add.

For scenarios where data should be editable, where a dataset cursor is important, where a very large dataset is used, ... it is recommended to use TDBAdvGrid with `PageMode` set to true.

TDBAdvGrid setup

To get started with TDBAdvGrid, the easiest way is to drop a TDataSource on the form and connect it to a TDataSet of choice. Activate the dataset and connect the TDataSource to TDBAdvGrid. With the default setting of TDBAdvGrid.AutoCreateColumns as true, the grid will automatically create TDBGridColumnItem instances for every DB field in the dataset and add it to the Columns collection. The TDBGridColumnItem.FieldName will be initialized with the fieldnames found in the dataset Fields collection. When TDBAdvGrid.AutoRemoveColumns is set to true, deactivating the dataset will automatically remove all columns from the grid.

The TDBGridColumn further offers following customizations for the grid:

Alignment	Sets
AutoMaxSize	Sets the maximum size of a column width during calls to grid.AutoColumnSize()
AutoMinSize	Sets the minimum size of a column width during calls to grid.AutoColumnSize()
BorderPen	Sets the pen for the cell borders in the column
Borders	Defines what borders of the cell to draw
CheckBoxField	When true, the field is treated as a Boolean field represented by a checkbox
CheckFalse	Sets the value of the DB field that corresponds with an unchecked checkbox
CheckTrue	Sets the value of the DB field that corresponds with a checked checkbox
Color	Sets the column background color
ColumnPopup	Sets the popup menu that is used for the column
ColumnPopupType	Defines when the popup menu should be displayed, ie. on left or right mouse click, for normal, fixed or all cells.
ComboItems	Holds possible values for a combobox inplace editor for the column
DataImageField	When true, the field value is used as index to display an image of the connected ImageList
EditLength	Sets the max. edit length for the inplace editor
EditLink	Sets the TEditLink component that is used to bind external editors as inplace editors for the grid.
EditMask	Sets the editmask for an inplace mask editor
Editor	Sets the editor type to be used for the column
FieldName	Sets the fieldname for the dataset field that is displayed in the column
Filter	Sets a list of filter values for a column in the grid
FilterCaseSensitive	When true, a filter specified for the column is treated as case sensitive
Fixed	When true, the column's cells are displayed as fixed cells
FloatFormat	Sets the format specifier for DB fields that are floating point data. For a list of possible format specifiers, see TAdvStringGrid developers guide or in the Borland Help file under the Format() function
Font	Sets the font of the column
Header	Sets the column header text. When empty, the FieldName.DisplayLabel is used as column header text
HeaderAlignment	Sets the alignment of the header text
HeaderFont	Sets the font of the header text
HTMLTemplate	Sets the HTML template of the column. This allows to display multiple DB fields in one column. See paragraph on using the HTMLTemplate
MaxSize	When different from zero, this is the max. width a column can have when columnsizing by user is enabled (with goColSizing = true in grid.Options)
MinSize	When different from zero, this is the min. width a column can have when columnsizing by user is enabled (with goColSizing = true in grid.Options)
Name	Sets the name of the TDBGridColumnItem instance
Password	When true, a password inplace editor is used and the field is displayed with

	the asterix character.
PictureField	When true and a blob field is connected to the column, the blob data is treated as an image and displayed
PrintBorderPen	Sets the pen used for printing the cell border
PrintBorders	Sets what borders of a cell to print
PrintColor	Sets the column background cell color for printing
PrintFont	Sets the column font for printing
ProgressBKColor	Sets the background color for a progressbar
ProgressColor	Sets the foreground color for a progressbar
ProgressField	When true, the field value in the column is treated as a value to set the progressbar. Note that this requires that the field value is between 0 and 100. If this is not available in the dataset, a calculated field could be used to generate a value between 0 and 100.
ProgressTextBKColor	Sets the text color on a progressbar for the background coloured part
ProgressTextColor	Sets the text color on a progressbar for the foreground coloured part
ReadOnly	When true, the column cannot be edited
ShowBands	When true, the column will display the bands as defined by the property grid.Bands
ShowUnicode	When true, the grid will try to display the blob field data as Unicode text
SortPrefix	Sets the prefix text part to ignore for internal sorting. If text has a currency specifier as prefix for example, this prefix can be ignored for sorting on the value.
SortSuffix	Sets the suffix text part to ignore for internal sorting. If text has a currency specifier as suffix for example, this suffix can be ignored for sorting on the value.
SpinMax	Sets the maximum value for a spinedit inplace editor
SpinMin	Sets the minimum value for a spinedit inplace editor
SpinStep	Sets the step value for a spinedit inplace editor
Tag	Integer value for custom use
Width	Sets the width of the column

Other than the column properties, some other settings need to be considered. With PageMode set to true, some datasets maintain internally an order for returning the pages of displayed rows to the grid and some not. This depends on the implementation of the TDataSet component that is used to connect to the database of choice. In general, when there is a problem with scrolling in the grid, it is recommended to set the property DataSetType to dtNonSequenced.

TDBAdvGrid also tries to know the number of rows in the dataset internally. To be database implementation independent, this is done by getting the result of the operation

```
Dataset.First
rowcount := DataSet.MoveBy($FFFFFFFF);
```

Some dataset implementations have a slow MoveBy operation that causes in turn that TDBAdvGrid is slow. A solution for this is to use a SQL 'SELECT COUNT' operation to supply the number of rows in the dataset to TDBAdvGrid via the event OnGetRecordCount. Most datasets have a fast implementation of the 'SELECT COUNT' command. With TDBAdvGrid.PageMode set to true and OnGetRecordCount implemented via a 'SELECT COUNT'. The sample application BDERecordCount in the TDBAdvGrid samples distribution shows how this can be done. In this sample, OnGetRecordCount is implemented as:

```
procedure TForm1.DBAdvGrid1GetRecordCount(Sender: TObject;
```

```

    var Count: Integer);
begin
    Query2.SQL.Text := 'select Count(*) from Country.db';
    Query2.Active := True;
    Count := Query2.Fields[0].AsInteger;
    Query2.Active := False;
end;

```

TDBAdvGrid sorting

TAdvStringGrid offers built-in automatic sorting by a click on the column header. This is an often required feature that can be used with TDBAdvGrid too. When the PageMode property is set to false, all data is available in the grid and it is ready to be sorted by the internal sorting mechanism of the grid. Other than setting SortSettings.Show to true, there is nothing to do to enable the sorting. When PageMode is set to true though, only visible records are loaded in the grid. The internal sorting of the grid can as such not be applied to sort a full dataset. It is required to perform a sort through the dataset itself. TDBAdvGrid has built-in code that detects whether a TQuery or TADOQuery dataset is connected and will automatically generate the property SQL ORDER BY clause to perform the sorting on the dataset. For other dataset types, it is required though to implement the grid OnCanSort event. In the TDBAdvGrid samples distribution, the demo BDESort shows how this can be done. The implementation of this OnCanSort event is:

```

procedure TForm1.DBAdvGrid1CanSort(Sender: TObject; ACol: Integer;
    var DoSort: Boolean);
var
    fldname:string;
begin
    DoSort := False; // disable internal sort

    // toggle sort order
    if dbadvgrid1.SortSettings.Direction = sdAscending then
        dbadvgrid1.SortSettings.Direction := sdDescending
    else
        dbadvgrid1.SortSettings.Direction := sdAscending;

    // get field name of the column clicked
    fldname := query1.FieldList.Fields[ACol - 1].FieldName;

    if pos(' ',fldname) > 0 then
        fldname := 'biolife.db.'+fldname+'';

    // add ORDER BY clause to the query
    query1.SQL.Text := 'select * from biolife.db ORDER BY '+fldname;

    if dbadvgrid1.SortSettings.Direction = sdDescending then
        query1.SQL.Text := query1.SQL.Text + ' DESC';

```

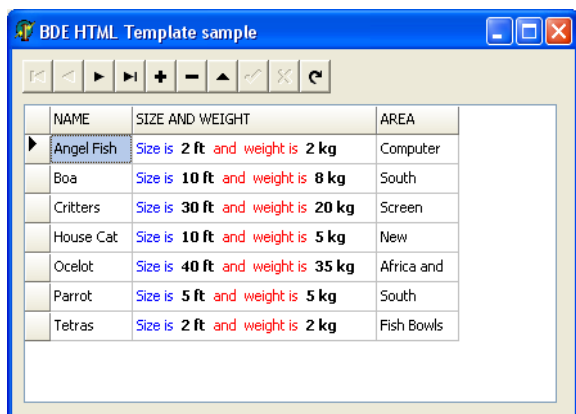
```
query1.Active := true;
DBAdvGrid1.SortSettings.Column := ACol;
end;
```

TDBAdvGrid HTML templates

Through a HTML template, it is possible to put multiple fields in a single cell with optionally HTML formatting. This is done by specifying a HTML template via the property DBAdvGrid.Columns[Index].HTMLTemplate. The HTML template is a string where a DB field reference set by <#DBFIELDNAME> will be replaced by the DB field value for each record for display. For example, when a table has a field SIZE and WEIGHT, the following HTML template creates a single cell with text in blue, red & bold:

```
DBAdvGrid.Columns[2].HTMLTemplate := '<FONT color="clBlue">Size is </FONT> <B><#SIZE> ft</B> <FONT color="clRed"> and weight is </FONT> <B><#Weight> kg</B>';
```

The resulting grid looks like:



NAME	SIZE AND WEIGHT	AREA
Angel Fish	Size is 2 ft and weight is 2 kg	Computer
Boa	Size is 10 ft and weight is 8 kg	South
Critters	Size is 30 ft and weight is 20 kg	Screen
House Cat	Size is 10 ft and weight is 5 kg	New
Ocelot	Size is 40 ft and weight is 35 kg	Africa and
Parrot	Size is 5 ft and weight is 5 kg	South
Tetras	Size is 2 ft and weight is 2 kg	Fish Bowls

TDBAdvGrid memo fields, Boolean fields & image blobs

TDBAdvGrid can automatically display memo fields, can show Boolean fields as checkboxes in the grid and can display images stored in blob fields. By default, a memo field is displayed as '(MEMO)' and an image blob is displayed as '(GRAPHIC)' just like in the Borland TDBGrid.

To show real contents of memo fields, set global property `DBAdvGrid.ShowMemoFields = true`. To have the cell sizes automatically sized according to the text in memo fields, it is sufficient to call `Grid.AutoSizeRows(false,4)` after activating the dataset.




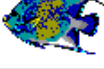

By default, a DB field of the type `ftBoolean` is displayed as value 'true' or 'false'. TDBAdvGrid can also automatically show such `ftBoolean` field type as a checkbox. To do this, set `DBAdvGrid.ShowBooleanFields = true`. One step further is that TDBAdvGrid can also show checkboxes for DB fields that do not have the type `ftBoolean` but that can have two values. This can be used with databases that do not support Boolean field types for example where a true condition is stored in a field as one value and false condition is stored as another value. To show checkboxes for such fields, set `DBAdvGrid.Columns[columnindex].CheckBoxField = true` and set via the properties `DBAdvGrid.Columns[columnindex].CheckTrue`, `DBAdvGrid.Columns[columnindex].CheckFalse` the values for a true condition and false condition.

Example:

Assume that an Interbase numeric field stores 1 as true condition and 0 as false condition. We want to display & edit this field with a checkbox. As the DB field type is `ftNumeric`, setting `DBAdvGrid.ShowBooleanFields` will not show this field as checkboxes. Setting `DBAdvGrid.Columns[numericfieldindex].CheckBoxField = true` will show checkboxes and the checked state of the checkbox is controlled with `DBAdvGrid.Columns[numericfieldindex].CheckTrue = '1'` and `DBAdvGrid.Columns[numericfieldindex].CheckFalse = '0'`.

	City	State	Zip	Country	Phone	FAX	Preferred
▶	Southfield	OH	60093	<input checked="" type="checkbox"/>	708-555-95	708-555-75	<input checked="" type="checkbox"/>
	Indianapolis	IN	46208	<input checked="" type="checkbox"/>	317-555-45		<input checked="" type="checkbox"/>
	Berkely	MA	02779	<input checked="" type="checkbox"/>	800-555-47	508-555-89	<input type="checkbox"/>
	Rancho	CA	90221	<input checked="" type="checkbox"/>	213-555-78		<input checked="" type="checkbox"/>
	Macon	GA	20865	<input checked="" type="checkbox"/>	912-555-67	912-555-84	<input type="checkbox"/>
	Redwood	CA	94065-1086	<input checked="" type="checkbox"/>	415-555-14	415-555-12	<input type="checkbox"/>
	Hapeville	GA	30354	<input checked="" type="checkbox"/>	800-555-62	404-555-82	<input checked="" type="checkbox"/>

TDBAdvGrid can also show BMP, GIF, JPEG images that are stored in BLOB fields. By default, such fields are displayed as '(GRAPHIC)'. When setting `DBAdvGrid.ShowPictureFields = true`, all fields with the type `ftGraphic` will be displayed as images. In some databases, the field type for an image field will not be `ftGraphic` but just `ftBlob` for example. The grid cannot automatically know that the BLOB data should be interpreted as an image though. If a DB field of the type `ftBlob` holds images, `DBAdvGrid.Columns[columnindex].PictureField` can be set to true and for this column, the grid will try to display the BLOB data as BMP, GIF or JPEG image.

	Length (cm)	Length_In	Notes	Graphic
▶	50	19.6850393 700787	Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and	
	60	23.6220472 440945	Called seaperch in Australia. Inhabits the areas around lagoon coral reefs and sandy bottoms.	
	229	90.1574803 149606	This is the largest of all the wrasse. It is found in dense reef areas, feeding on a wide variety of	
	30	11.8110236 220472	Habitat is around boulders, caves, coral ledges and crevices in shallow waters. Swims alone or in groups.	
	80	31.4960629 92126	Also known as the coronation trout. It is found around coral reefs from shallow to very deep	

TDBAdvGrid editing

TDBAdvGrid supports various inplace editors as well as the capability to use external controls as inplace editor for the grid. The TAdvStringGrid documentation has an overview of all possible inplace editors that are included and how to use external editors as well. To enable editing in the grid, it is required that

- the database table is editable
- the database field is editable
- DBAdvGrid.PageMode = true
- the option goEditing is set to true in DBAdvGrid.Options

TDBAdvGrid can perform editing in two modes. This is selected by the property DBAdvGrid.EditPostMode. When this is set to epCell, this means that the value that has been edited is posted immediately to the database when a cell leaves inplace editing. When DBAdvGrid.EditPostMode is set to epRow, this means that a Post will only happen when all cells in a row have been edited and the user moves to another row. Typically, for tables with various required fields, the epRow setting is the preferred setting.

When a selected cell is clicked, the inplace editing starts. The inplace editing can also be programmatically started and stopped with methods DBAdvGrid.ShowInplaceEdit, DBAdvGrid.HideInplaceEdit. It is important to know that editing of a cell or row stops when the inplace editor loses focus. Only when it loses focus, the edited value is either directly posted (epCell mode) or internally stored (epRow mode) to post when the row changes. In some circumstances, it is possible that a non focusable control performs an operation on the dataset that expected the editing to be finished. This is the case with the TDBNavigator. The grid cannot know that a TDBNavigator or other non focusable control is clicked. To avoid that the grid remains in editing mode when a database operation assumes editing has stopped, programmatically call DBAdvGrid.HideInplaceEdit. In case of the TDBNavigator, this could be done for example from the event DBNavigator.BeforeAction. Note that it is harmless that DBAdvGrid.HideInplaceEdit is called even when it is not in editing mode.

TDBAdvGrid Unicode support

TDBAdvGrid has built-in support to edit Unicode DB fields. Following Unicode inplace editors are available:

edUniComboEdit : Unicode editable combobox
edUniComboList : Unicode combobox
edUniEdit : regular Unicode inplace editor
edUniEditBtn : regular Unicode inplace editor with embedded button
edUniMemo : multiline Unicode memo editor

When the DB field type, returned by `TDBField.DataType` is `ftWideString`, TDBAdvGrid will automatically show the field as Unicode. Make sure though that a TrueType font is used that includes the Unicode charactersets that are being used. A very complete and recommend font is Arial Unicode MS.

In some databases that do not really support Unicode, it is possible that the Unicode is stored in a BLOB field. To show the Unicode properly in the grid, set `TDBAdvGrid.Columns[columnindex].ShowUnicode = true`. This setting will treat the binary data in the BLOB field as Unicode text.

Some additional helper events are available that can be used for Unicode editing:

`OnGetEditWideText` : this event is triggered just before inplace editing starts and allows to override the default text with which the inplace editor will be started.

`OnSetEditWideText` : this event is triggered when the inplace editing ends and returns the new text

`OnCellValidateWide` : this event is triggered when editing is about to end, it returns the new edited text and allows to modify, accept or cancel the edited text.