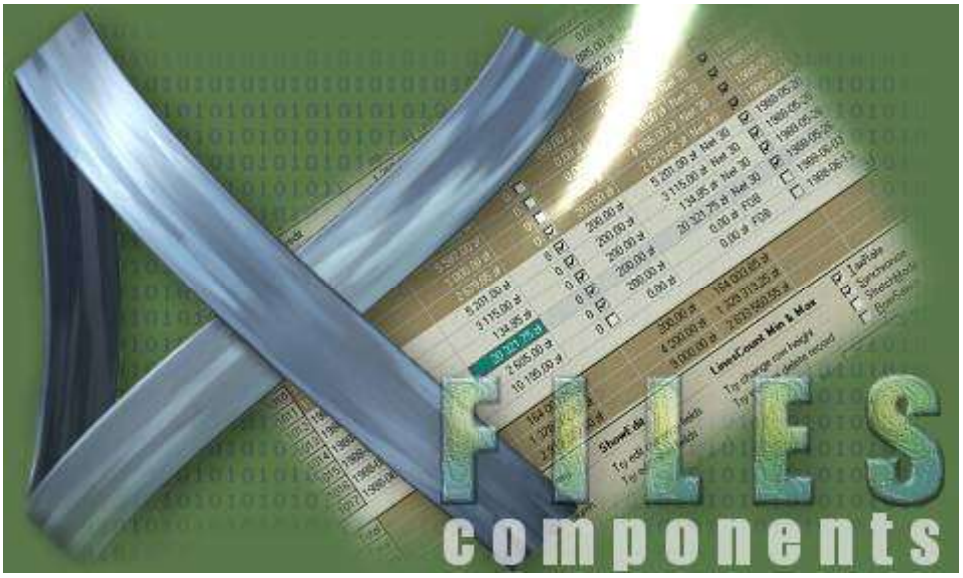
X-Files Software



Developer's Guide

Version 8.5

Table of Contents

Note. Find all occurrences of {* ver. 8.0 *} marker to read new topics only.

X-Files Components - Overview

X-Files Components 8.5 package contains a full set of professional advanced components designed for Delphi 5, 6, 7, 2005, 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11 Alexandria, 12 Athens, **13 Florence** and for C++Builder 5, 6, 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11 Alexandria, 12 Athens, **13 Florence**.
X-DBGrid Component 8.5 package contains few fundamental components designed for database developers. This package is a main part of X-Files Components package.

X-DBGrid Component package contains:

Main DB-Aware controls:

TXDBGrid component is a powerful functional extension of standard TDBGrid component. TXDBGrid component looks beautiful and implements many additional features like: fixed columns, Windows themes and custom styles, many variants of gradient drawing, proportional scrolling for any DataSet even filtered, expandable columns and title headers, checkboxes, hot buttons, title and indicator images, many variants of pictures drawing into cells, sorting markers and multicolumns sorting, enhanced lookup list, data list, calculator, calendar, storing and loading columns layout, enhanced support for bdRightToLeft BiDiMode, extended rows, columns and cells selection, auto-number and auto-select column, hints and tooltips, column stretching and many, many more ...

TXDBGrid offers in addition: auto-updated quick sequence number, auto-changed sorting order for all standard DataSet descendants (**ADO, BDE, CDS, DBX, IBX, FireDAC**) and many of third-party DAC components, auto-calculated totals values for whole DataSet and/or for SelectedRows, 4 kinds of integrated totals footers, extended lookup columns, universal search panel, integrated filtering system and auto-filter list for each column.

TXDBGrid is fully compatible with standard TDBGrid and can be used as a replacement without losing existing values. TXDBGrid has its own Column Editor and Designer, which can fully support all new column's properties.

TXDBColumn represents a standalone column editor. TXDBColumn performs the same functionality as inplace editor for the selected column in TXDBGrid component. TXDBColumn directly uses all column's properties set in TXDBGrid. The result of data editing in TXDBColumn is returned to the TXDBGrid.

TXDBEditor represents a standalone field editor. TXDBEditor performs the same functionality for a field in DataSet as inplace editor in TXDBGrid component. TXDBEditor allows to set properties similar to the column properties in TXDBGrid. The result of data editing in TXDBEditor is stored directly to the DataSet. When DataSource property is not assigned TXDBEditor can work as non db-aware editor.

TXDBColumnsDialog is a complementary component for TXDBGrid designed for grid columns management in run-time. The user can show/hide selected columns in the grid by using convenient dialog which presents columns list with checkboxes.

TXDBNavigator represents a gradient database navigator. TXDBNavigator is a functional extension of standard TDBNavigator. TXDBNavigator introduces several new properties to control border of buttons, gradient drawing style and parent's background.

DB-Aware graphic classes:

TXBlobField & TXGraphicField classes extend functionality of TBlobField & TGraphicField to achieve assignment compatibility between graphic BLOB field and TPicture class for another graphic formats. It allows directly using TImage, TDBImage, TQRImage, TQRDBImage with any graphic format stored in graphic BLOB fields (*.bmp, *.ico, *.gif, *.png, *.jpg and others supported by FieldGraphicClass).

Gradient graphic classes:

TXFGradient class is a fundamental class of gradient drawing style. It holds all properties and methods needed to realize gradient drawing for all controls. TXFGradient introduced a set of properties & options to select many variants of gradient drawing style. The properties, methods & events from this class allow to self-implement gradient drawing style in other controls.

X-Files Components package contains all above components/classes and the following:

DB-Aware reporting controls:

TXQRGrid is a complementary component for TXDBGrid designed for dynamic report creation on the basis of TXDBGrid current settings. TXQRGrid component can print and preview the content of TXDBGrid component. TXQRGrid creates in run-time complete report in QuickReport format. The report can be automatically stretched or created as few parts (vertical bands) depend on selected paper size settings.

TXDBPrintColumnsDialog is a complementary component for TXDBGrid & TXQRGrid designed for selection printed columns in run-time. User can select columns that are printed by using convenient dialog which presents columns list with checkboxes.

Additional DB-Aware controls:

TXDBText represents a data-aware control that displays the value of a field on a gradient background. TXDBText is derrived from the standard TDBText control, but Transparent property is default True.

TXDBCheckBox represents a data-aware transparent check box. TXDBCheckBox is derived from the standard TDBCheckBox control, but works effecive on gradient background.

TXDBRadioGroup represents a gradient group of radio buttons connected to a database. TXDBRadioGroup is similar to the standard TDBRadioGroup control, but introduces several properties to control gradient style and parent's background.

Gradient buttons:

TXFButton is a gradient push button control. TXFButton is derrived from the standard TButton control. TXFButton introduces several properties to control border and gradient style.

TXFBitBtn is a gradient bitmap button control. TXFBitBtn is derrived from the standard TBitBtn control. TXFBitBtn introduces several properties to control border and gradient style.

TXFSpeedButton is a gradient button control. TXFSpeedButton is derrived from the standard TSpeedButton control. TXFSpeedButton introduces several properties to control border and gradient style.

TXFUpDown is a gradient up-down control. TXFUpDown is similar to the standard TUpDown control, but introduces several properties to control border and gradient style.

Transparent controls:

TXFLabel is a nonwindowed control that displays text on a gradient background. TXFLabel is derrived from the standard TLabel control, but Transparent property is default True.

TXFStaticText is a windowed control that displays text on a gradient background. TXFStaticText is similar to the standard TStaticText control, but Transparent property works effective on a gradient background.

TXFCheckBox represents a transparent check box. TXFCheckBox is similar to the standard TCheckBox control, but works effecive on gradient background.

TXFRadioButton represents a transparent radio button. TXFRadioButton is similar to the standard TRadioButton control, but works effecive on gradient background.

Gradient panels:

TXFGroupBox represents a gradient group box. TXFGroupBox is similar to the standard TGroupBox control, but introduces several properties to control gradient style and parent's background.

TXFRadioGroup represents a gradient group of radio buttons that function together. TXFRadioGroup is similar to the standard TRadioGroup control, but introduces several properties to control gradient style and parent's background.

TXFCheckGroup represents a gradient group of check buttons that function together. TXFCheckGroup is similar to the TRadioGroup control, but introduces Checked property to maintain state of check buttons and several properties to control gradient style and parent's background.

TXFPanel represents a gradient panel. TXFPanel is similar to the standard TPanel control, but introduces several properties to control gradient style and parent's background.

Gradient bars:

TXFStatusBar represents a gradient status bar. TXFStatusBar is similar to the standard TStatusBar control, but introduces several properties to control gradient style.

TXFTrackBar represents a gradient track bar. TXFTrackBar is similar to the standard TTrackBar control, but introduces several properties to control gradient style.

TXFProgressBar represents a gradient progress bar. TXFProgressBar is similar to the standard TProgressBar control, but introduces several properties to control gradient style.

X-Files Components - Requirements

Delphi/C++Builder updates:

The package requires using latest released updates for each version of Delphi/C++Builder: Delphi 5.01, Delphi 6.02 RTL 3, Delphi 7.01, Delphi 2005.03 Win32, Delphi 2006.02 Win32, Delphi 2007.03 Win32, Delphi 2009.03, Delphi 2010.05, Delphi XE.01, Delphi XE2.04, Delphi XE3.02, Delphi XE4.01, Delphi XE5.02, Delphi XE6.01, Delphi XE7.01, Delphi XE8.01, Delphi 10 Seattle, Delphi 10.1 Berlin, Delphi 10.2 Tokyo, Delphi 10.3 Rio, Delphi 10.4 Sydney, Delphi 11 Alexandria, **12**, C++Builder 5.01, C++Builder 6.04, C++Builder 2006.02, C++Builder 2007.03, C++Builder 2009.03, C++Builder 2010.05, C++Builder XE.01, C++Builder XE2.04, C++Builder XE3.02, C++Builder XE4.01, C++Builder XE5.02, C++Builder XE6.01, C++Builder XE7.01, C++Builder XE8.01, C++Builder 10 Seattle, C++Builder 10.1 Berlin, C++Builder 10.2 Tokyo, C++Builder 10.3 Rio, C++Builder 10.4 Sydney, C++Builder 11 Alexandria, **12**, You may can't install the package on the Trial/Personal version of some Delphi/C++Builder.

QuickReport updates:

TXQRGrid component requires to install latest version of QuickReport Standard for appropriate version of Delphi/C++Builder. For Delphi 5 - QR 3.5 (qr35sd5.exe), for Delphi 6 - QR 3.5.1 (qr351sd6.exe), for Delphi 7 - QR 3.5.1 (qr351sd7.exe), for C++Builder 5 - QR 3.5 (qr35sc5.exe), for C++Builder 6 - QR 3.5 (qr35sc6.exe).

The latest version of QuickReport Standard may be downloaded from QuSoft AS Home Page or directly from X-Files Components Website. If you want to use TXQRGrid component with other version of QuickReport (e.g. Professional) you should register the Professional version of X-Files Components (with source) to recompile this package.

TXQRGrid component for Delphi 2005 and higher is not delivered in X-Files Components Standard package due the QuickReport Standard package is missing in these versions of Delphi. To can still use TXQRGrid component in Delphi 2005 or higher you must register X-Files Components Professional and QuickReport Professional. TXQRGrid component works correctly with latest versions of QuickReport Professional 3.6x, 4.0x, 5.0x.

Windows Themes:

To get (on Delphi/C++Builder 5 & 6) Windows Themes for TXDBGrid component and to use Windows Themes for other controls delivered in the package, you should download and install Windows XP Theme Manager (freeware by Mike Lischke) from http://www.soft-gems.net/ and register the Professional version (with source) of the package. (Mike Lischke is author of Themes unit delivered in Delphi 7 and higher).

Graphics in BLOB fields:

To can display additional graphic formats in TXDBGrid (*.pcx, *.scr, *.tif, *.eps, etc.) you should download GraphicEx library by Mike Lischke (Mozilla Public Licence 1.1) from http://www.soft-gems.net/ and register the Professional version (with source) of a package.

To can using TXBlobField & TXGraphicField as default classes for BLOB fields, you must register the Professional version (with source) of a package, to can recompile it.

X-Files Components - Installation

To install X-Files Components package you should:

1. Uninstall previous version of X-Files Components / X-DBGrid Component, if exists.
2. Run **X-FilesNNXXX.exe** installer.

To install X-DBGrid Component package you should:

1. Uninstall previous version of X-DBGrid Component / X-Files Components, if exists.
2. Run **X-DBGridNNXXX.exe** installer.

How to update Library Path ?

Make sure your **Library Path** contains only one path pointing to X-Files Components or X-DBGrid Component package.

- The default path for X-Files Components is:
  $(DELPHI)\XFiles - for Delphi 5, 6, 7
  $(BCB)\XFiles - for C++Builder 5, 6
  $(BDS)\XFiles - for Delphi 2005, 2006, 2007, 2009, 2010, XE, XE2, ..., XE8
  $(BDS)\XFiles - for C++Builder 2006, 2007, 2009, 2010, XE, XE2, ..., XE8
  $(BDS)\XFiles - for RAD Studio 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney
  $(BDS)\XFiles - for RAD Studio 11 Alexandria
- The default path for X-DBGrid Component is:
  $(DELPHI)\XDBGrid - for Delphi 5, 6, 7
  $(BCB)\XDBGrid - for C++Builder 5, 6
  $(BDS)\XDBGrid - for Delphi 2005, 2006, 2007, 2009, 2010, XE, XE2, ..., XE8
  $(BDS)\XDBGrid - for C++Builder 2006, 2007, 2009, 2010, XE, XE2, ..., XE8
  $(BDS)\XDBGrid - for RAD Studio 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney
  $(BDS)\XDBGrid - for RAD Studio 11 Alexandria

When you install the new package, the new installer can automatically change path in the **Library Path** for this reason.

How to install Context Help File ?

The installer can automatically install **Context Help File**, then you don't need to do it manualy. If your OpenHelp can't find any topic from **X-Files Components Reference** file, you should:

1. Select the **Help | Customize...** menu item to display **OpenHelp** window.
2. Add the **..\Help\XFiles.cnt** file to the **Contents** page.
3. Add the **..\Help\XFiles.hlp** file to the **Index** page.
4. Add the **..\Help\XFiles.hlp** file to the **Link** page.
5. Click on **File | Save Project** to save your changes.

Notice. You may need to remove any indexes you do not use because default help indexing is near the limits of Microsoft's WinHelp. If you experience an empty list on the Index tab page of WinHelp, this indicates you need to remove some files from OpenHelp.

How to install Context Help File for Delphi 2005 or higher ?

Delphi 2005 and higher uses help files in HtmlHelp format. You can't integrate WinHelp XFiles.hlp file with HtmlHelp, but you can still **open XFiles.hlp file from Start Menu Folder**, you have selected during installation.

How to replace TDBGrid component with TXDBGrid component in old projects ?

TXDBGrid is fully compatible with TDBGrid component on properties level and can be used as a replacement without losing existing values.

1. Replace "**DBGrids**" unit name to "**XDBGrids**" unit in uses clause.
2. Replace all occurence of "**TDBGrid**" class to "**TXDBGrid**" class.
3. Click right mouse button on a form and select "**View as Text**".
4. Replace all occurence of "**TDBGrid**" class to "**TXDBGrid**" class.
5. Click right mouse button on editor and select "**View as Form**".
6. Save changes, (important) **close the form and reopen it**.
7. Replace other class name to "**TX**" class name (e.g. **TColumn** to **TXColumn**).

How to replace standard controls with appropriate gradient controls in old projects ?

1. Check "**Install Demo files**" and "**Compile Demo files**" tasks during package installation.
2. Run **XFChanger** application from **Start Menu Folder**, you have selected during installation.
3. Select the unit(s) to exchange standard controls with gradient controls.

How to recompile X-DBGrid Component package ?

1. Open **Conditionals.pas** source file from **$(Delphi)\XDBGrid\Source** directory.
2. Change swich(es), save the file on disk and close Delphi/C++Builder environment.
3. Find **DCCXDBGridNN.bat** file in **$(Delphi)\XDBGrid\Source** directory and run it.
4. Check, the compilation for both packages finished successfully.
5. Press Enter to install recompiled files, press Ctrl+C to cancel installation.

To do this you should have the Professional version (with source) of the package.

How to recompile X-Files Components package ?

1. Open **Conditionals.pas** source file from **$(Delphi)\XFiles\Source** directory.
2. Change swich(es), save the file on disk and close Delphi/C++Builder environment.
3. Find **DCCXFilesNN.bat** file in **$(Delphi)\XFiles\Source** directory and run it.
4. Check, the compilation for both packages finished successfully.
5. Press Enter to install recompiled files, press Ctrl+C to cancel installation.

To do this you should have the Professional version (with source) of the package.

How to achieve using Windows Themes under Delphi/C++Builder 5 & 6 ?

To achieve using Windows Themes the TThemeServices class is required. This class is available in Themes unit as early as Delphi 7. To achieve themed version of all X-Files controls under Delphi/C++Builder 5 & 6, you should:

1. Download **Windows XP Theme Manager** by **Mike Lischke** (freeware) from http://www.soft-gems.net/ .
2. Open **ThemeManagerN.dpk** package and select "**Explicit rebuild**" Build control in Package Options.
3. Compile and install Windows XP ThemeManager package.
4. Copy **ThemeManagerN.dcp** file from **$(Delphi)\Projects\Bpl** directory to **$(Delphi)\XFiles\Source** directory.
5. Replace switch `{$UNDEF THEMES}` with `{$DEFINE THEMES}` in **..\Source\Conditionals.pas** file.
6. Recompile X-DBGrid Component or X-Files Components package.
7. Remember to add **XP Manifest** either file or resource to your project.

To do this you must have the Professional version (with source) of the package.

How to use X-Files Components Professional with QuickReport Professional ?

X-Files Components package is compiled with using latest version of **QuickReport Standard**. When the fatal error occurs (`[Fatal Error] Project1.dpr(5): Unit XQRGrids was compiled with a different version of QuickRpt.TCustomQuickRep`) you must recompile this package. To recompile package with other version of QuickReport (e.g. Professional), you should:

1. Copy QuickReport's run-time *.dcp file (e.g **QR4RunD7.dcp**) from **$(Delphi)\Projects\Bpl** to **$(Delphi)\XFiles\Source** directory.

1a. To recompile also 64-bit version of the package you must copy 64-bit version of QuickReport's run-time *.dcp file to $(Delphi)\XFiles\Source\Win64 directory.

2. Replace switch `{$UNDEF XQRGRID}` with `{$DEFINE XQRGRID}` in **..\Source\Conditionals.pas** file, if needed.
3. Define switch `QR36`, `QR40` or either `QR50` in **..\Source\Conditionals.pas** file, depend on QuickReport's version.
4. Check, the name of QuickReport's run-time package is correct in **..\Source\VCLXFilesNN.dpk** file.
5. Recompile X-Files Components package.

To do this you must have the Professional version (with source) of the package.

Notice. Some versions of **Quick Report Professional** have selected incorrect Build control switch in Package Options ("Rebuild as needed"). In that case, you must first change this option to "**Explicit rebuild**" and rebuild both packages of Quick Report, to successfully recompile X-Files Components Professional package.

How to display additional graphic formats (*.pcx, *.tiff, etc.) in TXDBGrid ?

TXDBGrid component can directly display *.bmp, *.ico, *.wmf, *.emf, *.gif, *.png & *.jpg graphics stored in BLOB fields, when DataType of BLOB field is ftGraphic. To directly display additional graphic formats in TXDBGrid, you should:

1. Download **GraphicEx** library by **Mike Lischke** (Mozilla Public Licence 1.1) from http://www.soft-gems.net/ .
2. Open **..\GraphicEx\GraphicConfiguration.inc** file and select graphic formats you need to include.
3. Open **..\Source\DCCXDBGridNN.bat** or **..\Source\DCCXFilesNN.bat** batch file and add path to **GraphicEx** library after **-u** switch (e.g. -u..\..\GraphicEx).
4. Replace switch `{$UNDEF GRAPHICEX}` with `{$DEFINE GRAPHICEX}` in **..\Source\Conditionals.pas** file.
5. Recompile X-DBGrid Component or X-Files Components package.

To do this you must have the Professional version (with source) of the package.

How to display additional graphic formats in TDBImage, TQRDBImage ?

TDBImage & TQRDBImage can display only *.bmp graphics stored in BLOB fields, because TBlobField & TGraphicField classes supports assignment only from/to TBitmap graphic class. To achieve assignment compatibility between graphic BLOB field and TPicture class for additional graphic formats (*.ico, *.wmf, *.emf, *.gif, *.png, *.jpg and others supported by **GraphicEx** library) you should to add XDBFields unit to the uses clause and replace TBlobField & TGraphicField class names with TXBlobField & TXGraphicField. You can do it manually or by running **XFChanger** application from **Start Menu Folder**, you have selected during installation.

How to install TXBlobField & TXGraphicField as default classes for BLOB fields ?

To install TXBlobField & TXGraphicField as default classes for BLOB fields, you should:

1. Replace switch `{$UNDEF XDBFIELDS}` with `{$DEFINE XDBFIELDS}` in **..\Source\Conditionals.pas** file.
2. Recompile X-DBGrid Component or X-Files Components package.

To do this you must have the Professional version (with source) of the package.

Notice. From now, all necessary BLOB fields added in Fields Editor will be created as TXBlobField & TXGraphicField, but you still must to replace TBlobField & TGraphicField class names with TXBlobField & TXGraphicField in existing *.dfm & *.pas files. You can do it manually or by running **XFChanger** application from **Start Menu Folder**, you have selected during installation.

X-Files Components - Support

X-Files Support

If you experience any technical problems, please feel free to contact me for support. I will try to answer your questions within 24 hours. Before you'll send e-mail, please check X-Files Components FAQ page for looking up a ready solution for your problem.

Sincerely,

Krzysztof Szyszka, X-Files Software
Developer of X-Files Components
Embarcadero Technology Partner

Website: http://www.x-files.pl/
E-mail: support@x-files.pl
E-mail: sales@x-files.pl
E-mail: news@x-files.pl

Troubleshooting

1. Using TXDBGrid component on Middle Eastern locale.

TXDBGrid component implements bdRightToLeft BiDiMode on the platform whether the local represents a Middle Eastern locale (SysLocale.MiddleEast = True), but some bugs have been detected in antecesor class (TCustomGrid).

To ensure correct working for TXDBGrid component when SysLocale.MiddleEast is True and BiDiMode is bdRightToLeft, you need fix the bugs in Grids.pas unit and add source of the Grids unit to your project. These are the statements from Grids.pas unit, you need to improve:

```
{ Need for Delphi 5, 6, 7, 2005, 2006, 2007, 2009, 2010, XE, XE2 }{ Solved in
XE3 }

function TCustomGrid.CalcCoordFromPoint(X, Y: Integer;
  const DrawInfo: TGridDrawInfo): TGridCoord;
...
begin
  if not UseRightToLeftAlignment then
    Result.X := DoCalc(DrawInfo.Horz, X)
  else
    Result.X := DoCalc(DrawInfo.Horz, ClientWidth - X);  // Added by K.S.
12/23/2000
//  Result.X := DoCalcRightToLeft(DrawInfo.Horz, X); // Commented by K.S.
12/23/2000
  Result.Y := DoCalc(DrawInfo.Vert, Y);
end;
```



```
{ Need for Delphi 5 only }

procedure TCustomGrid.CalcSizingState(X, Y: Integer; var State: TGridState;
  var Index: Longint; var SizingPos, SizingOfs: Integer;
  var FixedInfo: TGridDrawInfo);

  procedure CalcAxisState(const AxisInfo: TGridAxisDrawInfo; Pos: Integer;
    NewState: TGridState);
  var
    I, Line, Back, Range: Integer;
  begin
    if NewState = gsColSizing then // Added by K.S. 12/23/2000
    if UseRightToLeftAlignment then
      Pos := ClientWidth - Pos;


{ Need for Delphi 5 only }

procedure TCustomGrid.MouseDown(Button: TMouseButton; Shift: TShiftState;
  X, Y: Integer);
...
    if FGridState <> gsNormal then
    begin
      if FGridState = gsColSizing then // Added by K.S. 12/23/2000
      if UseRightToLeftAlignment then
        FSizingPos := ClientWidth - FSizingPos;
      DrawSizingLine(DrawInfo);
      Exit;
    end;


{ Need for Delphi 5 only }

procedure TCustomGrid.MouseUp(Button: TMouseButton; Shift: TShiftState;
  X, Y: Integer);
...
      gsRowSizing, gsColSizing:
        begin
          CalcDrawInfo(DrawInfo);
          DrawSizingLine(DrawInfo);
          if FGridState = gsColSizing then // Added by K.S. 12/23/2000
          if UseRightToLeftAlignment then
            FSizingPos := ClientWidth - FSizingPos;
```

If you'll implement these changes, mouse click on fixed columns and row resizing by the user should work correct for TXDBGrid component and for each TCustomGrid's descendant when SysLocale.MiddleEast is True and BiDiMode is bdRightToLeft.

2. Using TXDBGrid component when invisible column starts from XDBGrid.ClientWidth.

When the property Visible will be False for the column that starts from XDBGrid.ClientWidth the Invalidate method work incorrect. When you'll click on several cells, you may see old and new selection for this cells. You need fix the bug in Grids.pas unit and add source of this unit to your project. These are the statements from Grids.pas unit, you need to improve:

```
{ Need for Delphi 5, 6, 7, 2005, 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4,
XE5, XE6, ... }

procedure TCustomGrid.GridRectToScreenRect(GridRect: TGridRect;
  var ScreenRect: TRect; IncludeLine: Boolean);

  function LinePos(const AxisInfo: TGridAxisDrawInfo; Line: Integer): Integer;
...
      for I := Start to Line - 1 do
      begin
        Inc(Result, GetExtent(I) + EffectiveLineWidth);
        if Result > GridExtent + EffectiveLineWidth then // Added by K.S.
09/25/2002
//      if Result > GridExtent then // See CalcAxis  // Commented by K.S.
09/25/2002
        begin
          Result := 0;
          Exit;
        end;
```

DISCLAIMER OF WARRANTY

THIS SOFTWARE IS PROVIDED TO YOU "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED INCLUDING BUT NOT LIMITED TO THE APPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE. YOU ASSUME THE ENTIRE RISK AS TO THE ACCURACY AND THE USE OF THE SOFTWARE AND ALL OTHER RISK ARISING OUT OF THE USE OR PERFORMANCE OF THIS SOFTWARE AND DOCUMENTATION.

X-Files Software SHALL NOT BE LIABLE FOR ANY DAMAGES WHATSOEVER ARISING OUT OF THE USE OF OR INABILITY TO USE THIS SOFTWARE, EVEN IF X-Files Software HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL X-Files Software BE LIABLE FOR ANY CONSEQUENTIAL, NCIDENTAL, DIRECT, INDIRECT, SPECIAL, PUNITIVE, OR OTHER DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO DAMAGES OR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS, EVEN IF X-Files Software HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TXDBGrid

TXDBGrid displays and manipulates records from a dataset in a tabular grid.

Unit
XDBGrids

Description
TXDBGrid component is a powerful functional extension of standard TDBGrid component.

TXDBGrid component implements many additional features like: fixed columns, Windows themes and custom styles, many variants of gradient drawing, proportional scrolling for any DataSet even filtered, expandable columns and title headers, checkboxes, hot buttons, title and indicator images, many variants of pictures drawing into cells, sorting markers and multicolumns sorting, enhanced lookup list, data list, calculator, calendar, storing and loading columns layout, enhanced support for bdRightToLeft BiDiMode, extended rows, columns and cells selection, auto-number and auto-select column, hints and tooltips, column stretching and many, many more ...

TXDBGrid offers in addition: auto-updated quick sequence number, auto-changed sorting order for all standard DataSet descendants (**ADO, BDE, CDS, DBX, IBX, FireDAC**) and many of third-party DAC components, auto-calculated totals values for whole DataSet and/or for SelectedRows, 4 kinds of integrated totals footers, extended lookup columns, integrated filtering system and auto-filter list for each column.

TXDBGrid is fully compatible with standard TDBGrid and can be used as a replacement without losing existing values. TXDBGrid has its own Column Editor and Designer, which can fully support all new column's properties.

TXDBGrid implements the generic behavior introduced in TXCustomDBGrid. TXDBGrid publishes many of the properties inherited from TXCustomDBGrid, but does not introduce any new behavior.

See also: TDBGrid, TXQRGrid

TXCustomDBGrid

TXCustomDBGrid is the abstract base class for grid controls that display the records from a dataset in a tabular format.

Unit
XDBGrids

Description
TXCustomDBGrid introduces new properties, events, and methods to expand the capabilities of TCustomDBGrid.

Do not create instances of TXCustomDBGrid. Use TXCustomDBGrid as a base class when declaring grid objects that display information from datasets. Properties and methods of TXCustomDBGrid provide basic behavior that descendant classes inherit as well as behavior that components can override to customize their behavior.

See also: TCustomDBGrid, TXDBGrid

TXCustomDBGrid.AlignStretchToBorder

Specifies whether the stretched grid is aligned to border frame.

```
property AlignStretchToBorder: Boolean; {* ver. 6.0 *}
```

Description
Use AlignStretchToBorder property to determine whether the grid is aligned to border frame. AlignStretchToBorder returns True when StretchMode is active, vertical scrollbar is not visible, grid is not themed and control has not 3D border frame.

AlignStretchToBorder is a read-only property.

See also: TXCustomDBGrid.AlignTotalsToBorder

TXCustomDBGrid.AlignTotalsToBorder

Specifies whether the stretched grid is aligned to border frame.

```
property AlignTotalsToBorder: Boolean; {* ver. 6.0 *}
```

Description
Use AlignTotalsToBorder property to determine whether the grid is aligned to border frame. AlignTotalsToBorder returns True when any total row is visible, horizontal scrollbar is not visible, grid is not themed and control has not 3D border frame.

AlignTotalsToBorder is a read-only property.

See also: TXCustomDBGrid.AlignStretchToBorder

TXCustomDBGrid.BlankHeight

Specifies the height (in pixels) of blank row.

```
property BlankHeight: Integer;
```

Description
If the OptionsEx property include dgBlankRow, BlankHeight specifies a height of blank row between data rows and total rows, otherwise BlankHeight specifies a height of blank area between data rows and the bottom of grid. BlankHeight is calculated as ClientHeight - TitleHeight - TotalHeight - DataRowsHeight.

BlankHeight is a read-only property.

See also: TXCustomDBGrid.TitleHeight, TXCustomDBGrid.TotalHeight, TXCustomDBGrid.DataRowsHeight, TXCustomDBGrid.DataHeight

TXCustomDBGrid.BorderStyle

Determines whether a single line border is drawn around the grid.

```
property BorderStyle: TBorderStyle;
```

Description
Set BorderStyle to bsSingle to add a single line border around the grid's image. Set BorderStyle to bsNone to omit the border. When BorderStyle is bsSingle and Ctl3D property is False, the color of line border is automatically adjusted to current FixedStyle property. It allows to obtain flat or semi-flat look for the grid control.

See also: TXCustomDBGrid.FixedStyle, TXCustomDBGrid.ListBorder, TXCustomDBGrid.Ctl3D

TXCustomDBGrid.ColLineWidth

Specifies the width (in pixels) of the lines that separate the columns of the grid.

```
property ColLineWidth: Integer;
```

Description
If the Options property include dgColLines, ColLineWidth is 1, otherwise ColLineWidth is 0.

ColLineWidth is a read-only property.

See also: TXCustomDBGrid.RowLineWidth, TXCustomDBGrid.ScrollBarWidth, TXCustomDBGrid.IndicatorWidth

TXCustomDBGrid.ColsChanged

Indicates the changes were made in SelectedCols list.

```
property ColsChanged: Boolean; {* ver. 4.3 *}
```

Description
The ColsChanged flag indicates the changes were made in SelectedCols list after BeginSelect method was called at the start of the changes. When the changes were made the ColsChanged is True. The ColsChanged flag is cleared by EndSelect method.

ColsChanged is a read-only property

See also: TXCustomDBGrid.BeginSelect, TXCustomDBGrid.EndSelect, TXCustomDBGrid.OnSelectedColsChanged

TXCustomDBGrid.ColsSelected

Specifies whenever any column or data cell is multiselected.

```
property ColsSelected: Boolean; {* ver. 4.3 *}
```

Description
Read ColsSelected to determine is any column or data cell currently multiselected in the grid. ColsSelected returns True when ColsSelection is True and SelectedCols list is not empty, otherwise ColsSelected return False.

ColsSelected is a read-only property.

See also: TXCustomDBGrid.MultiSelected, TXCustomDBGrid.RowsSelected, TXCustomDBGrid.ColsSelection, TXCustomDBGrid.SelectedCols

TXCustomDBGrid.ColsSelection

Specifies whenever any column or data cell is multiselected.

```
property ColsSelection: Boolean; {* ver. 4.3 *}
```

Description
Read ColsSelection to determine is multiselection of columns or cells currently active. ColsSelection returns True when MultiSelect is msCols or msCells, otherwise ColsSelection return False. When ColsSelection is True, the SelectedCols list contains selected columns (msCols) or columns of selected area (msCells).

ColsSelection is a read-only property.

See also: TXCustomDBGrid.RowsSelection, TXCustomDBGrid.MultiSelect, TXCustomDBGrid.SelectedCols, TXCustomDBGrid.ColsSelected

TXCustomDBGrid.ColumnNames

Describes the display attributes of the columns.

```
property ColumnNames[const FieldName: string]: TXColumn read GetColumnNames;
default; {* ver. 4.31 *}
property ColumnNames[const FieldName: WideString]: TXColumn read
GetColumnNames; default; // Delphi 2006, 2007 Win32, 2009, 2010, XE
```

Description
Use ColumnNames to read or set attributes of the columns in TXDBGrid. The ColumnNames property allows to access to Columns collection with using FieldName as an index of the collection. The ColumnNames is default property for TXDBGrid. It allows you to use short syntax `XDBGrid1[FieldName]` instead `XDBGrid1.ColumnNames[FieldName]`.

Example:

```
  CustomerDBGrid['Country'].Visible := False;
  Text := CustomerDBGrid['Company'].TotalHeader.DisplayText;
```

See also: TXCustomDBGrid.Columns

TXCustomDBGrid.Columns

Describes the display attributes and field bindings of the columns.

```
property Columns: TXDBGridColumns;
```

Description
Use Columns to read or set the field bindings and display attributes of the columns in TXDBGrid. Columns is an indexed collection of TXColumn objects. Use the properties of the TXColumn objects to specify the display attributes or field bindings of individual columns in the grid. The field binding of a column designates a field within the dataset specified by the DataSource property.

See also: TXColumn, TXDBGridColumns, TXCustomDBGrid.ColumnNames

TXCustomDBGrid.ColumsWidth

Specifies the width (in pixels) of all visibled columns in the grid.

```
property ColumnsWidth: Integer;
```

Description
The ColumnsWidth property determines the width of all visibled columns in the grid in pixels. When ColumnsWidth is greater than ClientWidth the horizontal scrollbar appears in the grid.

ColumnsWidth is a read-only property.

See also: TXCustomDBGrid.ScrollBarWidth, TXCustomDBGrid.FixedColsWidth, TXCustomDBGrid.IndicatorWidth

TXCustomDBGrid.Ctl3D

Determines whether a grid has a three-dimensional (3-D) or two-dimensional look.

```
property Ctl3D: Boolean;
```

Description
When BorderStyle property is bsSingle the Ctl3D property determines whether the grid has a flat (or semi-flat) or beveled appearance.

See also: TXCustomDBGrid.FixedStyle, TXCustomDBGrid.BorderStyle, TXCustomDBGrid.Ctl3DAuto

TXCustomDBGrid.Ctl3DAuto

Determines whether a Ctl3D property is automatically adjusted to current FixedStyle/FixedTheme property.

```
property Ctl3DAuto: Boolean;
```

Description
When Ctl3DAuto property is True the Ctl3D property is automatically changed when FixedStyle and/or FixedTheme property is changed to assume the best look for current grid style. When FixedStyle is fsDefault or fsSoft the Ctl3D property is True (three-dimensional look), otherwise the Ctl3D property is False (flat or semi-flat look). When FixedTheme is active (IsGridThemed return True) then Ctl3D property is always changed to True. When Ctl3DAuto property is False the Ctl3D property leaves unchanged.

See also: TXCustomDBGrid.FixedStyle, TXCustomDBGrid.BorderStyle, TXCustomDBGrid.Ctl3D

TXCustomDBGrid.CurrentCol

Specifies "proxy" class that returns enumerator for current column in the grid.

```
property CurrentCol: TXDBGridCurrentCol; {* ver. 6.2 *} // For Delphi 2009 or higher
```

Description

Use CurrentCol only when you need enumerator for current column in the grid.

```
type
  Column: TXColumn;
begin
  for Column in XDBGrid1.CurrentCol do
  begin
    // Perform action on current column into XDBGrid1
  end;
end;
```

See also: TXCustomDBGrid.CurrentRow, TXDBGridColumns.GetEnumerator, TXBookmarkList.GetEnumerator

TXCustomDBGrid.CurrentRow

Specifies "proxy" class that returns enumerator for current data row in the grid.

```
property CurrentRow: TXDBGridCurrentRow; {* ver. 6.2 *} // For Delphi 2009 or higher
```

Description

Use CurrentRow only when you need enumerator for current data row in the grid.

```
type
  Bookmark: TBookmark;
begin
  for Bookmark in XDBGrid1.CurrentRow do
  begin
    // Perform action on the current record into XDBGrid1.DataSource.DataSet
  end;
end;
```

See also: TXCustomDBGrid.CurrentCol, TXDBGridColumns.GetEnumerator, TXBookmarkList.GetEnumerator

TXCustomDBGrid.DataColLineColor

Specifies the color of the data column lines in the grid

```
property DataColLineColor: TColor; {* ver. 4.3 *}
```

Description
Set DataColLineColor to specify custom color for the data column lines in the grid.

The DataColLineColor property can handle two special colors: clDefault, clNone.
Select clDefault value to use DefaultDataLineColor value (clSilver).
Select clNone value when the data column lines should not be drawed.

See also: TXCustomDBGrid.DataRowLineColor, TXCustomDBGrid.FixedLineColor, TXCustomDBGrid.DefaultDataLineColor

TXCustomDBGrid.DataHeight

Specifies the height (in pixels) of data area.

```
property DataHeight: Integer;
```

Description
DataHeight is calculated as ClientHeight - TitleHeight - TotalHeight.

DataHeight is a read-only property.

See also: TXCustomDBGrid.DataRowsHeight, TXCustomDBGrid.BlankHeight

TXCustomDBGrid.DataLeftCol

Specifies the index of the first visible scrollable column in the grid.

```
property DataLeftCol: Integer;
```

Description
Read DataLeftCol to determine the index of the first column in the scrollable region that is visible. Set DataLeftCol to scroll the columns in the grid so that the column with index DataLeftCol is the first column after the fixed columns.

See also: TXCustomDBGrid.FixedCols

TXCustomDBGrid.DataRowCount

Indicates the number of data rows shown by the grid.

```
property DataRowCount: Integer;
```

Description
Examine DataRowCount to determine the number of data rows shown by the grid. Applications might use this property with Position to iterate through all visibled data rows in the grid. When DataLink is active, DataRowCount return value of RecordCount, otherwise DataRowCount return 0.

DataRowCount is a read-only property.

See also: TXCustomDBGrid.GotoPosition, TXCustomDBGrid.Position, TXCustomDBGrid.TitleRowCount, TXCustomDBGrid.TotalRowCount

TXCustomDBGrid.DataRowLineColor

Specifies the color of the data row lines in the grid

```
property DataRowLineColor: TColor; {* ver. 4.3 *}
```

Description
Set DataRowLineColor to specify custom color for the data row lines in the grid.

The DataRowLineColor property can handle two special colors: clDefault, clNone.
Select clDefault value to use DefaultDataLineColor value (clSilver).
Select clNone value when the data row lines should not be drawed.

See also: TXCustomDBGrid.DataColLineColor, TXCustomDBGrid.FixedLineColor, TXCustomDBGrid.DefaultDataLineColor

TXCustomDBGrid.DataRows

Specifies "proxy" class that returns enumerator for all data rows in the grid.

```
property DataRows: TXDBGridDataRows; {* ver. 5.6 *} // For Delphi 2009 or higher
```

Description
Use DataRows only when you need enumerator for all data rows in the grid.

```
type
  Bookmark: TBookmark;
begin
  for Bookmark in XDBGrid1.DataRows do
  begin
    // Perform action on each record into XDBGrid1.DataSource.DataSet
  end;
end;
```

See also: TXDBGridColumns.GetEnumerator, TXBookmarkList.GetEnumerator

TXCustomDBGrid.DataRowsHeight

Specifies the height (in pixels) of all data rows.

```
property DataRowsHeight: Integer;
```

Description
DataRowsHeight is calculated as sum of height of all data rows visibled in the grid.

DataRowsHeight is a read-only property.

See also: TXCustomDBGrid.TitleHeight, TXCustomDBGrid.TotalHeight, TXCustomDBGrid.BlankHeight, TXCustomDBGrid.DataHeight

TXCustomDBGrid.DefaultWidthMax

Specifies the maximum width for default columns.

```
property DefaultWidthMax: Integer; {* ver. 6.3 *}
```

Description
Set DefaultWidthMax to the maximum value of column's width for default columns. The DefaultWidthMin and DefaultWidthMax properties define the initial range for Width property for default columns. DefaultWidthMax is useful for very large string fields.

See also: TXCustomDBGrid.DefaultWidthMin

TXCustomDBGrid.DefaultWidthMin

Specifies the minimum width for default columns.

```
property DefaultWidthMin: Integer; {* ver. 6.3 *}
```

Description
Set DefaultWidthMin to the minimum value of column's width for default columns. The DefaultWidthMin and DefaultWidthMax properties define the initial range for Width property for default columns.

See also: TXCustomDBGrid.DefaultWidthMax

TXCustomDBGrid.DetailFields

Identifies detail fields in the detail dataset linked to the grid.

```
property DetailFields: string; {* ver. 6.2 *}
property DetailFields: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
DetailFields specifies detail fields in master->detail relationship when detail dataset is linked to the grid. When you specify this property the ChangeIndexFieldsand CurrentIndexFields methods will automatically take account of DetailFields when master->detail relationship is active.

See also: TXCustomDBGrid.OrderFields, TXCustomDBGrid.KeyFields

TXCustomDBGrid.DoubleBuffered

Determines whether the grid's image is rendered directly to the window or painted to an in-memory bitmap first.

```
property DoubleBuffered: Boolean;
```

Description
When DoubleBuffered is False, the grid paints itself directly to the window. When DoubleBuffered is True, the grid paints itself to an in-memory bitmap that is then used to paint the window. Double buffering reduces the amount of flicker when the grid repaints, but the rows scroll speed is slowest. Set DoubleBuffered to False only when you must run your application on slow machines. Especially, you should leave DoubleBuffered = True, when StripeColor is not clNone, or FixedTheme is not ftNone or the user can resize the grid in run-time.

See also: TWinControl.PaintWindow

TXCustomDBGrid.DragRows

Determines using drag-and-drop operations for grid's data row(s).

```
property DragRows: Boolean;
```

Description
Set DragRows to True (for DragKind = dgDrag and DragMode = dmManual), to set whether the XDBGrid should initial drag-and-drop operations for grid's data row(s).

See also: TControl.DragCursor, TControl.DragKind, TControl.DragMode

TXCustomDBGrid.DrawingOptions

Specifies the drawing options of the some color's properties.

```
type
  TGridDrawingOption = (gdoHighlightText, gdoDataColLines, gdoDataRowLines,
    gdoSelectedCell, gdoSelectedRow, gdoMultiSelected, gdoTotalSelected,
    gdoStripedRows, gdoEditorColor);
  TGridDrawingOptions = set of TGridDrawingOption;
property DrawingOptions: TGridDrawingOptions; {* ver. 5.1 *}
```

Description
Use DrawingOptions to easily include/exclude using of some Color's properties. The DrawingOptions is a master property for other published properties of TColor type. When a drawing option is included, the corresponding Color property is set to clDefault. When a drawing option is excluded, the corresponding Color property is set to clNone. The current value of DrawingOptions always shows which Color's properties are efficient used (are different than clNone). These are the possible values of DrawingOptions:

Value	Meaning
gdoHighlightText	The HighlightText property is used (highlighted text is visible).
gdoDataColLines	The DataColLinesColor property is used (lines of columns are drawn).
gdoDataRowLines	The DataRowLinesColor property is used (lines of rows are drawn).
gdoSelectedCell	The SelectCellColor property is used (focused cell is highlighted).
gdoSelectedRow	The SelectRowColor property is used (current row is highlighted).
gdoMultiSelected	The SelectionColor property is used (multiselected cells are highlighted).
gdoTotalSelected	The Totals.SelectedColor property is used (selected totals are highlighted).

gdoStripedRows	The StripeColor property is used (striped rows are visible).
gdoEditorColor	The EditorColor property is used (edited cell has changing color).

See also: TXCustomDBGrid.DrawingStyle

TXCustomDBGrid.DrawingStyle

Specifies the drawing style of the fixed cells.

```
type
  TGridDrawingStyle = (gdsClassic, gdsThemed, gdsGradient); // For
    C++Builder/Delphi 5, 6, 7, 8, 2005, 2006, 2007, 2009
  TGridDrawingStyle = Grids.TGridDrawingStyle;              // For
    C++Builder/Delphi 2010 or higher
property DrawingStyle: TGridDrawingStyle; {* ver. 5.0 *}
```

Description
Use DrawingStyle to select drawing style for fixed cells. The DrawingStyle is a master property for FixedStyle, FixedTheme and Gradient property. These are the possible values of DrawingStyle:

Value	Meaning
gdsClassic	The classic drawing style dependent on current FixedStyle value.
gdsThemed	The themed drawing style dependent on current FixedTheme value.
gdsGradient	The gradient drawing style dependent on current Gradient settings.

See also: TXCustomDBGrid.FixedStyle, TXCustomDBGrid.FixedTheme, TXCustomDBGrid.Gradient

TXCustomDBGrid.EditorColor

Specifies the background color of the editor and editor's list in the grid.

```
property EditorColor: TColor;
```

Description
Set EditorColor to specify background color for editor in the grid.

The EditorColor property can handle two special colors: clDefault, clNone.
Select clDefault value to use DefaultEditorColor value (clInfoBk).
Select clNone value to leave editor color unchanged.

See also: TXCustomDBGrid.SelectCellColor, TXCustomDBGrid.SelectionColor, TXCustomDBGrid.SelectRowColor, TXCustomDBGrid.StripeColor

TXCustomDBGrid.FillerButton

Determines using title of indicator as a button.

```
property FillerButton: Boolean;
```

Description
If the Options property include dgTitleButtons, set FillerButton to True to use title of indicator (Filler) as button.

See also: TXColumnTitle.Button, TXCustomDBGrid.FillerColor, TXCustomDBGrid.FillerHint, TXCustomDBGrid.FillerIndex, TXCustomDBGrid.FillerPopupMenu

TXCustomDBGrid.FillerColor

Specifies the background color of the title of indicator in the grid.

```
property FillerColor: TColor;
```

Description
Set FillerColor to specify custom color for the title of indicator (Filler) in the grid.

See also: TXCustomDBGrid.HeaderColor, TXCustomDBGrid.FillerButton, TXCustomDBGrid.FillerHint, TXCustomDBGrid.FillerIndex, TXCustomDBGrid.FillerPopupMenu

TXCustomDBGrid.FillerHint

Contains the text string that can appear when the user moves the mouse over the title of indicator.

```
property FillerHint: string;
```

Description
See HintOptions and TControl.Hint to read more.

See also: TXCustomDBGrid.FillerButton, TXCustomDBGrid.FillerColor, TXCustomDBGrid.FillerIndex, TXCustomDBGrid.FillerPopupMenu, TXCustomDBGrid.ONCellHint

TXCustomDBGrid.FillerIndex

Specifies which image is displayed in the title of indicator.

```
property FillerIndex: Integer;
```

Description
Use the FillerIndex property with the TitleImages property to specify the image for the title of indicator (Filler). If the FillerIndex is less than -1, then FillerIndex specifies image from Indicators calculated as -FillerIndex - 2.

See also: TXCustomDBGrid.FillerButton, TXCustomDBGrid.FillerColor, TXCustomDBGrid.FillerHint, TXCustomDBGrid.FillerPopupMenu

TXCustomDBGrid.FillerPopupMenu

Identifies the pop-up menu associated with the title of indicator.

```
property FillerPopupMenu: TPopupMenu;
```

Description
Assign a value to FillerPopupMenu to make a pop-up menu appear when the user selects the title of indicator (Filler) and clicks the right mouse button.

See also: TXCustomDBGrid.IndicatorPopupMenu, TXCustomDBGrid.FillerButton, TXCustomDBGrid.FillerColor, TXCustomDBGrid.FillerHint, TXCustomDBGrid.FillerIndex, TXColumnTitle.PopupMenu

TXCustomDBGrid.FillerToolMenu

Determines using ToolMenu as FillerPopupMenu.

```
property FillerToolMenu: Boolean; {* ver. 8.0 *}
```

Description
Set FillerToolMenu to True to use ToolMenu as FillerPopupMenu.

See also: TXCustomDBGrid.FillerPopupMenu, TXCustomDBGrid.PopupToolMenu, TXCustomDBGrid.ToolMenu

TXCustomDBGrid.Filter

Contains the Filter condition from DataSet linked to TXDBGrid.

```
property Filter: string; {* ver. 6.0 *}
```

Description
Filter property points to TDataSet.Filter condition stored in DataSet linked to TXDBGrid. When the Filter property is changed in TXDBGrid the new Filter string is stored in DataSet property, but the change is controlled by TXDBGrid. If KeyFields are defined, the current record is restored, if the record is still available in filtered record set. If dgAutoKeepSelection option is included in OptionsEx and grid has multiselected records, still visible selected records are restored.

See also: TXCustomDBGrid.Filtered, TXCustomDBGrid.OnFilterRecord

TXCustomDBGrid.Filtered

Specifies whether or not filtering is active for a dataset.

```
property Filtered: Boolean; {* ver. 6.0 *}
```

Description
Filtered property points to TDataSet.Filtered state of DataSet linked to TXDBGrid. When the Filtered property is changed in TXDBGrid the Filtered state is changed in DataSet, but the change is controlled by TXDBGrid. If KeyFields are defined, the current record is restored, if the record is still available in filtered record set. If dgAutoKeepSelection option is included in OptionsEx and grid has multiselected records, still visible selected records are restored.

See also: TXCustomDBGrid.Filter, TXCustomDBGrid.OnFilterRecord

TXCustomDBGrid.FilterGrid

Indicates the TXDBGridFilter that determines internal filter settings for the grid.

```
property FilterGrid: TXDBGridFilter;
```

Description
The FilterGrid property points to TXDBGridFilter object that determines internal filter settings for the grid.

See also: TXDBGridFilter

TXCustomDBGrid.FixedCols

Specifies the number of columns on the left of the grid that cannot be scrolled.

```
property FixedCols: Integer;
```

Description
See TCustomGrid.FixedCols to read more.

See also: TXCustomDBGrid.FixedStyle

TXCustomDBGrid.FixedColsWidth

Specifies the width (in pixels) of all fixed columns in the grid.

```
property FixedColsWidth: Integer;
```

Description
The FixedColsWidth property determines the width of all fixed columns in the grid in pixels.

FixedColsWidth is a read-only property.

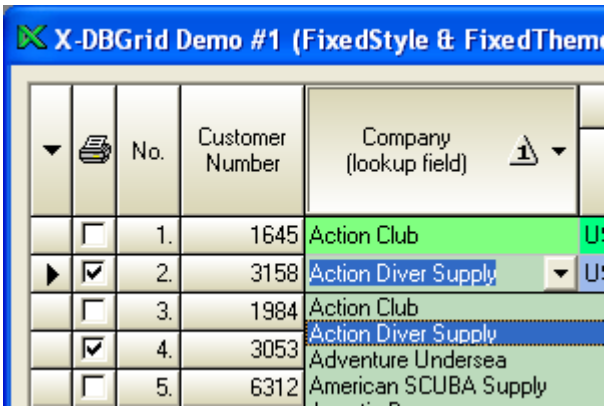
See also: TXCustomDBGrid.ScrollBarWidth, TXCustomDBGrid.ColumnsWidth, TXCustomDBGrid.IndicatorWidth

TXCustomDBGrid.FixedStyle

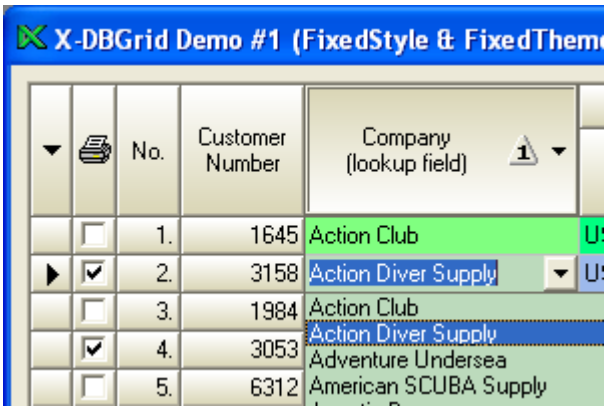
Specifies the style of the fixed cells in grid.

```
type
  TFixedStyle = (fsDefault, fsSoft, fsNice, fsFlat, fsGreat, fsGray, fsFine,
    fsMild);
property FixedStyle: TFixedStyle;
```

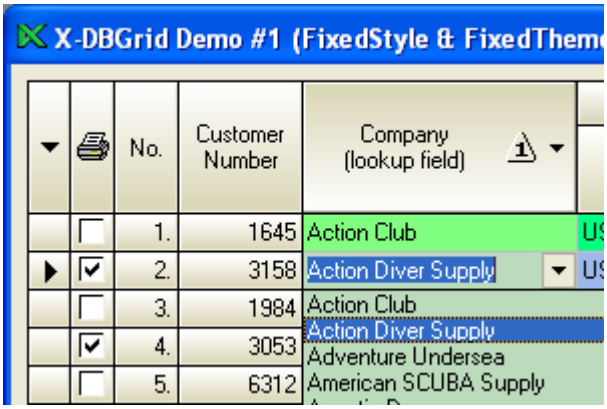
Description
Use FixedStyle to draw a variety lines that appears as a frame around fixed cells. The FixedStyle is a master property for other design properties like: Ctl3D, HotButtons, ListBorder, MarkerStyle, CheckBoxStyle, ScrollProp.Style. These are the possible values of FixedStyle:



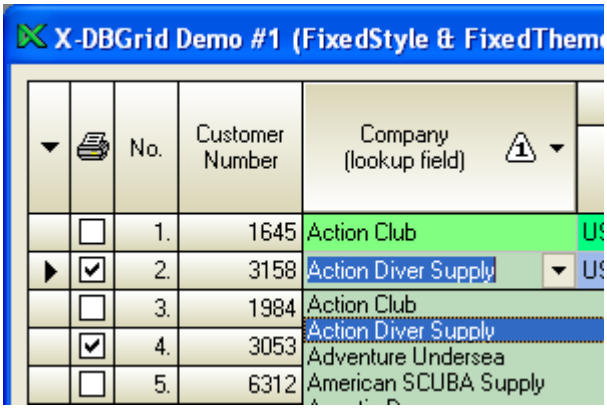
fsDefault



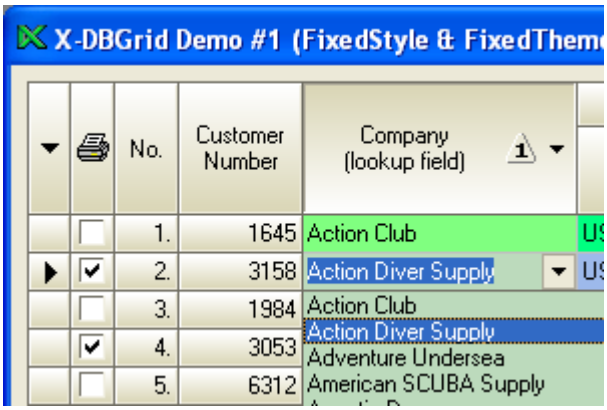
fsSoft



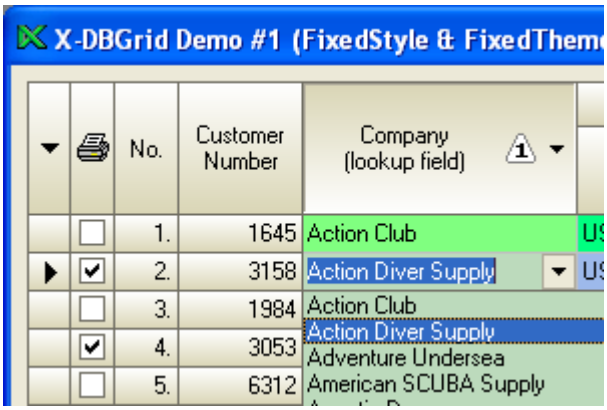
fsNice



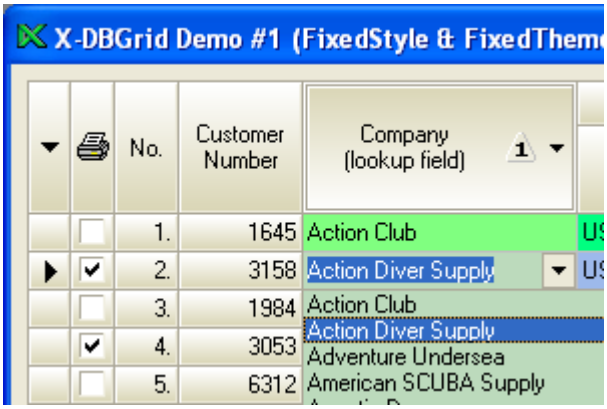
fsFlat



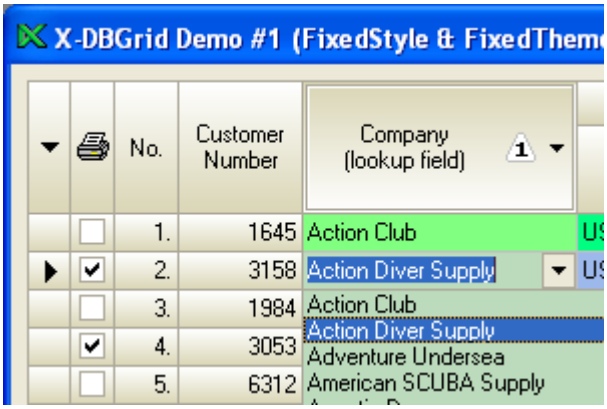
fsGreat



fsGray



fsFine



fsMild

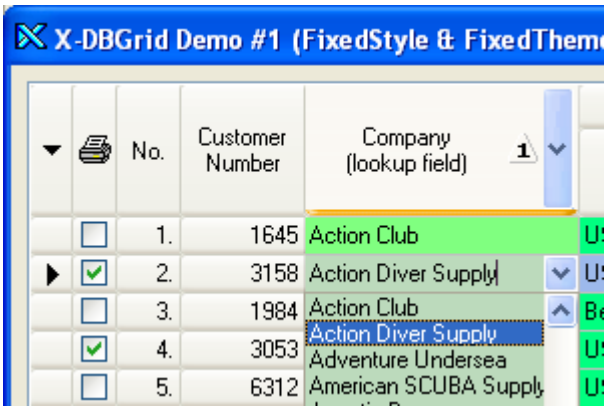
See also: TXCustomDBGrid.FixedCols, TXCustomDBGrid.FixedTheme, TXCustomDBGrid.BorderStyle, TXCustomDBGrid.Ctl3D

TXCustomDBGrid.FixedTheme

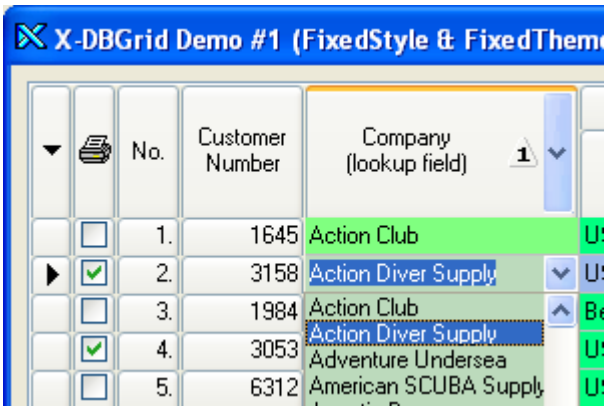
Specifies the theme of the fixed cells in grid (Windows XP only).

```
type
  TFixedTheme = (ftNone, ftDefault, ftButtons, ftTabs);
property FixedTheme: TFixedTheme;
```

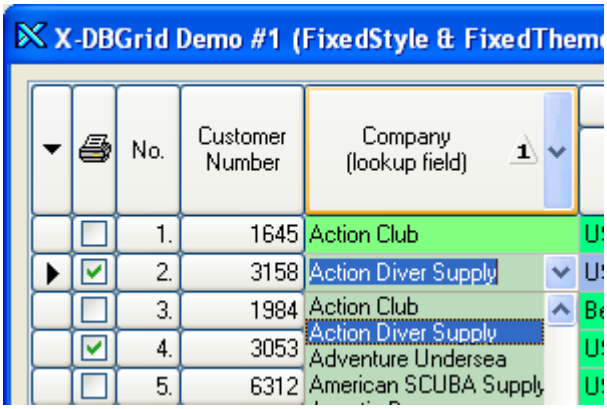
Description
Use FixedTheme to draw a variety elements that appears as a background of fixed cells. These are the possible values of FixedTheme:



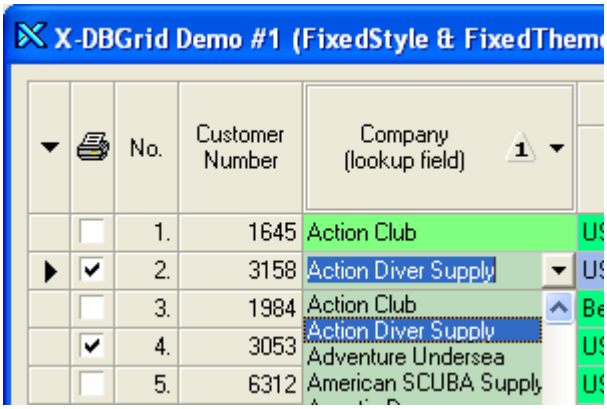
ftDefault



ftTabs



ftButtons



ftNone

Themed cells appear only on Windows XP platform when XP Manifest is present and dgColLines & dgRowLines are included in Options. Select ftNone value to disable themes on Windows XP platform.

Notice. The FixedTheme property requires TThemeServices class. This class is available only in Delphi 7 in Themes unit. To really use FixedTheme in earlier version of Delphi/C++Builder you should download and install **Windows XP Theme Manager** by **Mike Lischke** (freeware) from http://www.soft-gems.net/ and recompile XDBGrids.pas unit with using switch `{$DEFINE THEMES}`. To do this you should have the Professional version (with source) of the package.

See also: TXCustomDBGrid.FixedStyle

TXCustomDBGrid.FixedLineColor

Specifies the color of the fixed lines in the grid (lines between fixed cells).

```
property FixedLineColor: TColor; {* ver. 4.3 *}
```

Description
Read FixedLineColor property to get current color of fixed lines in the grid. The color of fixed lines depend on current FixedStyle and FixedTheme property.

The FixedLineColor is a readonly property.

See also: TXCustomDBGrid.DataColLineColor, TXCustomDBGrid.DataRowLineColor

TXCustomDBGrid.FlatSBMode

Indicates whether grid uses flat scrollbars.

```
function FlatSBMode: Boolean;
```

Description
Call FlatSBMode to determine whether XDBGrid uses flat scrollbars. FlatSBMode returns True when ComCtl32.dll supports flat scrollbars and when using of flat scrollbars is possible. Using of flat scrollbars is possible only when ThemesEnabled returns False and UseRightToLeftScrollBar return False.

When FlatSBMode is True the additional ScrolProp's possibilities are practicable.

See also: TXCustomDBGrid.ScrollProp, TXCustomDBGrid.ScrollBars

TXCustomDBGrid.FocusRect

Specifies kind of rectangle drawn around focused cell.

```
type
  TFocusRect = (frAuto, frDefault, frGray, frMild, frNone);
property FocusRect: TFocusRect;
```

Description
Use FocusRect to select a kind of rectangle drawn around focused cell. The FocusRect property is practicable for both non-Windows XP and Windows XP styles. These are the possible values of FocusRect:

Value	Meaning
frAuto	The focus rectangle is dependent on current FixedStyle and FixedTheme values.
frDefault	The focus rectangle is a dotted rectangle (system default).
frGray	The focus rectangle is drawn in gray color.
frMild	The focus rectangle is drawn in light gray color.
frNone	The focus rectangle isn't drawn.

See also: TXCustomDBGrid.FixedStyle, TXCustomDBGrid.FixedTheme

TXCustomDBGrid.Gradient

Specifies the gradient drawing style for the fixed cells in the grid.

```
property Gradient: TXDBGridGradient;
```

Description
The Gradient property points to TXDBGridGradient object that determines additional parameters needed to drawing gradient fixed and data cells. The gradient drawing style IsActive for grid only when not IsGridThemed.

See also: TXFGradient

TXCustomDBGrid.GridLineWidth

Specifies the width of the lines in the grid

```
property GridLineWidth: Integer; {* ver. 6.4 *}
```

Description
Set GridLineWidth to specify custom width for the lines in the grid. Default value is 1.

See also: TXCustomDBGrid.DataColLineColor, TXCustomDBGrid.DataRowLineColor

TXCustomDBGrid.GridStyle

Indicates the TXGridStyle that determines spacing for standard and XP drawing style.

```
property GridStyle: TXGridStyle;
```

Description
The GridStyle property points to TXGridStyle object that determines additional parameters needed to determine drawing style.

See also: TXCustomDBGrid.FixedStyle, TXCustomDBGrid.FixedTheme, TXGridStyle

TXCustomDBGrid.HeaderColor

Specifies the background color of the headers of columns in the grid.

```
property HeaderColor: TColor;
```

Description
Set HeaderColor to specify custom color for the headers of columns in the grid. HeaderColor is only meaningful if the Options property includes dgTitleHeaders.

See also: TXCustomDBGrid.FillerColor, TXCustomDBGrid.HeaderFont, TXCustomDBGrid.HeaderLinesCount

TXCustomDBGrid.HeaderFont

Describes the font used to draw the headers of columns in the grid.

```
property HeaderFont: TFont;
```

Description
Set HeaderFont to change the font used to draw the headers of columns. HeaderFont is only meaningful if the Options property includes dgTitleHeaders.

See also: TXCustomDBGrid.HeaderColor, TXCustomDBGrid.HeaderLinesCount

TXCustomDBGrid.HeaderLinesCount

Specifies the value that appears in the multi-line headers of columns in the grid.

```
property HeaderLinesCount: Integer;
```

Description
Set HeaderLinesCount to specify number of lines for multi-line headers of columns in the grid. HeaderLinesCount is only meaningful if the Options property includes dgTitleHeaders.

See also: TXCustomDBGrid.LinesCount, TXCustomDBGrid.TitleLinesCount, TXCustomDBGrid.HeaderColor, TXCustomDBGrid.HeaderFont

TXCustomDBGrid.HighlightText

Specifies the color of text for focused or multiselected cells in the grid.

```
property HighlightText: TColor; {* ver. 4.3 *}
```

Description
Set HighlightText to change color of text for focused or multiselected cells in the grid.

The HighlightText property can handle two special colors: clDefault, clNone.
Select clDefault value to use DefaultHighlightText value (clHighlightText).
Select clNone value to leave unchanged color of text (Font.Color).

See also: TXCustomDBGrid.SelectCellColor, TXCustomDBGrid.SelectionColor, TXCustomDBGrid.DefaultHighlightText

TXCustomDBGrid.HintOptions

Specifies various hint and tooltips properties of the XDBGrid.

```
type
  TXHintOption = (hoTitleHints, hoShowTitleHints, hoEditorHints,
    hoShowEditorHints, hoShowToolTips, hoKeepShowToolTips,
    hoAlwaysShowToolTips, hoDataHints, hoShowDataHints, hoAllowForGraphics,
    hoTotalHints, hoShowTotalHints, hoShowTotalToolTips);
  TXHintOptions = set of TXHintOption;
property HintOptions: TXHintOptions;
```

Description
Set HintOptions to include various hint and tooltips properties for the XDBGrid. Set ShowHint property to True to activate HintOptions. HintOptions is a set drawn from the following values:

Value	Meaning
hoTitleHints	Activates hints for individual titles and headers. Hints are specified in the Hint, HeaderHint and FillerHint properties.
hoShowTitleHints	Appears hint window for individual titles and headers when HintOptions includes hoTitleHints.
hoEditorHints	Activates hints for edited cells in the grid. Hints are specified in the EditorHint property.
hoShowEditorHints	Appears hint window for edited cells in the grid when HintOptions includes hoEditorHints.
hoShowToolTips	Appears tooltips window for longest text in the cell.
hoKeepShowToolTips	Tooltips window keeps visibility to the user moves mouse outside the cell. By default tooltips window is hidden after 2 x HintHidePause.
hoAlwaysShowToolTips	Appears tooltips window for any text in the cell when HintOptions includes hoShowToolTips.
hoDataHints	Activates hints for individual data cells. Hints are specified in the Hint and IndicatorHint properties. To activate hint for data cell you must exclude hoShowToolTips option from HintOptions or set ToolTips property to False for selected columns.
hoShowDataHints	Appears hint window for individual data cells when HintOptions includes hoDataHints.
hoAllowForGraphics	Appears hint or tooltips window for data cells also for graphics (Images or ftGraphic field). You may need to write OnColGetText event handler for ftGraphic field to replace default [GRAPHIC] text.
hoTotalHints	Activates hints for individual total cells. Hints are specified in the Hint property. To activate hint for total cell you must exclude hoShowTotalToolTips option from HintOptions or set ToolTips property to False for selected total cell.
hoShowTotalHints	Appears hint window for individual total cells when HintOptions includes hoTotalHints.
hoShowTotalToolTips	Appears tooltips for longest text in the total cell. The options hoShowToolTips and hoKeepShowToolTips are also applicable for hoShowTotalToolTips.

See also: TXCustomDBGrid.Options, TXCustomDBGrid.OnCellHint, TApplication.OnHint

TXCustomDBGrid.HotButtons

Specifies kind of hot buttons of the column titles.

```
type
  THotButtons = (hbAuto, hbNone, hbBump, hbTile, hbLowered, hbRaised);
property HotButtons: THotButtons;
```

Description
Use HotButtons to select a kind of style for title buttons when the mouse cursor moves over the title cell. The HotButtons property is practicable only for non-Windows XP styles when dgHotButtons option is included in Options. These are the possible values of HotButtons:

Value	Meaning
hbAuto	The style of hot buttons is dependent on current FixedStyle value.
hbNone	The hot buttons are unavailable for non-Windows XP styles but they are still available for Windows XP styles when dgHotButtons option is included in Options.
hbBump	The bumped hot buttons are most suitable for 3D title buttons (FixedStyle in [fsDefault, fsSoft]).
hbTile	The tiled hot buttons are most suitable for tiled title buttons (FixedStyle in [fsFine, fsMild]).
hbLowered	The hot buttons are similar to pressed buttons. There is default option for FixedStyle = fsNice.
hbRaised	The hot buttons are similar to flat's hot buttons. There is default option for FixedStyle = fsFlat.

See also: TXCustomDBGrid.Options, TXCustomDBGrid.FixedStyle

TXCustomDBGrid.IndicatorHint

Contains the text string that can appear when the user moves the mouse over the indicator.

```
property IndicatorHint: string;
```

Description

See HintOptions and TControl.Hint to read more.

See also: TXCustomDBGrid.FillerHint, TXCustomDBGrid.IndicatorPopupMenu, TXCustomDBGrid.IndicatorWidth, TXCustomDBGrid.OnCellHint

TXCustomDBGrid.IndicatorImages

Specifies custom images that are drawn in the indicator.

```
const {ImageIndex constants for IndicatorImages}
  iiClear = -1;
  iiArrow = 0;
  iiEdit = 1;
  iiInsert = 2;
  iiMultiArrow = 3;
  iiMultiDot = 4;
  iiMenu = 5;
  iiAnchor = 6;
  iiMultiAnchor = 7;
  iiArrowUp = 8;
  iiArrowDown = 9;
  iiArrowLeft = 10;
  iiArrowRight = 11;
  iiCheck = 12;
  iiCross = 13;
  iiSquare = 14;
  iiCircle = 15;
  iiTriangle = 16;
  iiPlus = 17;
  iiMinus = 18;
  iiExclamation = 19;
  iiFilter = 20; {* ver. 6.0 *}
  iiActiveFilter = 21; {* ver. 6.0 *}
  iiPrev = 22; {* ver. 6.2 *}
  iiNext = 23; {* ver. 6.2 *}
  iiFirst = 24; {* ver. 6.2 *}
  iiLast = 25; {* ver. 6.2 *}
  iiClose = 26; {* ver. 6.2 *}

property IndicatorImages: TImageList;
```

Description

Use IndicatorImages to provide a customized list of bitmaps that can be displayed in the indicator. You may change and/or expand default indicator's markers. Use OnCalcImageIndex event to select expanded indicator's markers.

See also: TXCustomDBGrid.TitleImages, TXCustomDBGrid.Indicators, TXCustomDBGrid.IndicatorWidth, TXCustomDBGrid.OnCalcImageIndex, TXColumn.Images

TXCustomDBGrid.IndicatorPopupMenu

Identifies the pop-up menu associated with the indicator.

```
property IndicatorPopupMenu: TPopupMenu;
```

Description
Assign a value to IndicatorPopupMenu to make a pop-up menu appear when the user selects the indicator and clicks the right mouse button.

See also: TXCustomDBGrid.FillerPopupMenu, TXColumnTitle.PopupMenu

TXCustomDBGrid.Indicators

Specifies the image list that are drawn in the indicator.

```
property Indicators: TImageList;
```

Description
If the IndicatorImages property is assigned, Indicators specifies IndicatorImages, otherwise Indicators specifies internal image list that are drawn in the indicator.

Markers is a read-only property.

See also: TXCustomDBGrid.Markers, TXCustomDBGrid.IndicatorImages, TXCustomDBGrid.IndicatorWidth

TXCustomDBGrid.IndicatorWidth

Specifies the width of the indicator.

```
property IndicatorWidth: Integer;
```

Description
The IndicatorWidth property determines the width of the indicator in pixels.

Notice. Since version 3.3 the default value for IndicatorWidth is 20 (instead 11). This value is suitable for new default VisualStyle.

See also: TXCustomDBGrid.Indicators, TXCustomDBGrid.Indicator.Images, TXCustomDBGrid.ColLineWidth, TXCustomDBGrid.ScrollBarWidth

TXCustomDBGrid.InUpdateListItems

Indicates the current operating mode of the grid.

```
property InUpdateListItems: Boolean; {* ver. 4.3 *}
```

Description
Use InUpdateListItems to determine the grid is currently during UpdateListItems.

InUpdateListItems is a read-only property.

See also: TXCustomDBGrid.UpdateListItems, TXCustomDBGrid.UpdateInProgress

TXCustomDBGrid.InUpdateSequence

Indicates the current operating mode of the grid.

```
property InUpdateSequence: Boolean;
```

Description
Use InUpdateSequence to determine the grid is currently during UpdateSequence.

InUpdateSequence is a read-only property.

See also: TXCustomDBGrid.UpdateSequence, TXCustomDBGrid.UpdateInProgress

TXCustomDBGrid.InUpdateTotals

Indicates the current operating mode of the grid.

```
property InUpdateTotals: Boolean;
```

Description
Use InUpdateTotals to determine the grid is currently during UpdateTotals.

InUpdateTotals is a read-only property.

See also: TXCustomDBGrid.UpdateTotals, TXCustomDBGrid.UpdateInProgress

TXCustomDBGrid.KeyFields

Identifies the key field or fields in the dataset linked to the grid.

```
property KeyFields: string;
property KeyFields: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
KeyFields specifies the field or fields which allows to identify any record in dataset linked to the grid. When you specify this property the ChangeDataSetOrder or RefreshDataSet methods will automatically restore current record. When KeyFields is empty, the first record in dataset will be current after calling this methods. KeyFields must be defined in format appropriate to use as a parameter for Locate method.

See also: TXCustomDBGrid.DetailFields, TXCustomDBGrid.OrderFields, TXCustomDBGrid.ChangeDataSetOrder, TXCustomDBGrid.RefreshDataSet

TXCustomDBGrid.LastShiftState

Indicates last state of the Alt, Ctrl, and Shift keys and the mouse buttons.

```
property LastShiftState: TShiftState;
```

Description
LastShiftState hold TShiftState flags last used by key-event and mouse-event handlers to determine the state of the Alt, Ctrl, and Shift keys and the state of the mouse buttons when the last event occurred.

LastShiftState is a read-only property.

See also: TXColumnTitle.ToggleMarker

TXCustomDBGrid.LinesCount

Specifies the value that appears in the multi-line rows of data in the grid.

```
property LinesCount: Integer;
```

Description
Set LinesCount to specify number of lines for multi-line rows of data in the grid. If the dgRowResize flag is set in the data grid's Options property, users can resize the row at runtime.

See also: TXCustomDBGrid.HeaderLinesCount, TXCustomDBGrid.TitleLinesCount, TXCustomDBGrid.LinesCountMin, TXCustomDBGrid.LinesCountMax

TXCustomDBGrid.LinesCountMax

Specifies the maximum value for LinesCount property.

```
property LinesCountMax: Integer;
```

Description
Set LinesCountMax to the maximum value the LinesCount property can take. The LinesCountMin and LinesCountMax properties define the total range over which LinesCount can't vary.

See also: TXCustomDBGrid.LinesCountMin, TXCustomDBGrid.LinesCount

TXCustomDBGrid.LinesCountMin

Specifies the minimum value for LinesCount property.

```
property LinesCountMin: Integer;
```

Description
Set LinesCountMin to the minimum value the LinesCount property can take. The LinesCountMin and LinesCountMax properties define the total range over which LinesCount can't vary.

See also: TXCustomDBGrid.LinesCountMax, TXCustomDBGrid.LinesCount

TXCustomDBGrid.ListBorder

Specifies kind of list border for pick list and lookup list.

```
type
  TListBorder = (lbAuto, lbDefault, lbGray, lbMild);
property ListBorder: TListBorder;
```

Description
Use ListBorder to select a kind of list border for pick and lookup lists when the cell is edited. The ListBorder property is practicable for both non-Windows XP and Windows XP styles. These are the possible values of ListBorder:

Value	Meaning
lbAuto	The style of list border is dependent on current FixedStyle value.
lbDefault	The list border is drawn in default black color suitable for FixedStyle in [fsDefault, fsNice, fsFlat].
lbGray	The list border is drawn in gray color suitable for all FixedStyle and FixedTheme values.
lbMild	The list border is drawn in light gray color suitable for FixedStyle in [fsFine,

fsMild] and all FixedTheme values.

See also: TXColumn.PickList, TXCustomDBGrid.FixedStyle, TXCustomDBGrid.FixedTheme

TXCustomDBGrid.ManageGrid

Specifies first XDBGrid in chain managed by this master grid.

```
property ManageGrid: TXCustomDBGrid;
```

Description
Use ManageGrid property to read, which XDBGrid has set this grid in MasterGrid property.

ManageGrid is a read-only property.

See also: TXCustomDBGrid.MasterGrid, TXCustomDBGrid.Synchronize

TXCustomDBGrid.Markers

Specifies the image list that are drawn as sorting markers in the titles.

```
property Markers: TImageList;
```

Description
If the TitleMarkers property is assigned, Markers specifies TitleMarkers, otherwise Markers specifies internal image list that are drawn as sorting markers.

Markers is a read-only property.

See also: TXCustomDBGrid.Indicators, TXCustomDBGrid.TitleMarkers, TXCustomDBGrid.MarkerStyle

TXCustomDBGrid.MarkerStyle

Specifies style of markers drawn in the titles.

```
type
  TMarkerStyle = (msAuto, msAutoGray, msDefault, msSoft, msRaisedDark,
    msRaisedGray, msRaisedMild, msLoweredDark, msLoweredGray, msLoweredMild,
    msFlat, msGray, msMild);
property MarkerStyle: TMarkerStyle;
```

Description
Use MarkerStyle to select a style of sorting markers drawn in the titles. The MarkerStyle property is practicable for both non-Windows XP and Windows XP styles but only when TitleMarkers property is undefined. These are the possible values of MarkerStyle:

Value	Meaning
msAuto	The style of markers is dependent on current FixedStyle and FixedTheme values.
msAutoGray	The style of markers is one from Gray styles dependent on current FixedStyle and FixedTheme values.
msDefault	The style of markers is 3D - suitable for FixedStyle = fsDefault.
msSoft	The style of markers is 3D - suitable for FixedStyle = fsSoft.
msRaisedDark	The style of markers is convex - suitable for FixedStyle = fsNice.
msRaisedGray	The style of markers is convex - suitable for all FixedStyle values.
msRaisedMild	The style of markers is convex - suitable for FixedStyle = fsFine.
msLoweredDark	The style of markers is concave - suitable for FixedStyle = fsNice.

msLoweredGray	The style of markers is concave - suitable for all FixedStyle values.
msLoweredMild	The style of markers is concave - suitable for FixedStyle = fsFine.
msFlat	The style of markers is flat - suitable for FixedStyle = fsFlat.
msGray	The style of markers is flat - suitable for all FixedStyle values.
msMild	The style of markers is flat - suitable for FixedStyle = fsMild.

See also: TXCustomDBGrid.FixedStyle, TXCustomDBGrid.FixedTheme, TXCustomDBGrid.MarkerTransparent, TXCustomDBGrid.Markers

TXCustomDBGrid.MarkerTransparent

Specifies background of markers drawn in the titles.

```
type
  TMarkerTransparent = (mtAuto, mtFalse, mtTrue);
property MarkerTransparent: TMarkerTransparent;
```

Description
Use MarkerTransparent to select a background of sorting markers drawn in the titles. The MarkerTransparent property is practicable for both non-Windows XP and Windows XP styles but only when TitleMarkers property is undefined. These are the possible values of MarkerTransparent:

Value	Meaning
mtAuto	The background of markers is dependent on current MarkerStyle value.
mtFalse	The background of markers is white.
mtTrue	The background of markers is transparent.

See also: TXCustomDBGrid.MarkerStyle, TXCustomDBGrid.Markers

TXCustomDBGrid.MasterGrid

Specifies master XDBGrid to link with corresponding columns in this grid to establish a relationship between the grids.

```
property MasterGrid: TXCustomDBGrid;
```

Description
Set MasterGrid property to specify the master XDBGrid to use in establishing a relationship between this grid and the one specified in MasterGrid.

Each time the column's width, order or visibility in the master grid changes, the new values are used to change corresponding columns in this grid.

See also: TXCustomDBGrid.ManageGrid, TXCustomDBGrid.Synchronize

TXCustomDBGrid.MultiSelect

Determines current kind of multiselection.

```
type
  TMultiSelection = (msNone, msRows, msCols, msCells, msFixed);
  TMultiSelect = msNone .. msCells;
property MultiSelect: TMultiSelect; {* ver. 4.3 *}
```

Description
The MultiSelect property determines the current kind of multiselection in the grid. The possible kinds of multiselection are limited by MultiSelectOptions property. The accepted value for MultiSelect must be included in MultiSelectOptions. In other case the MultiSelect property returns msNone.

The default value of MultiSelect is msRows (when msRows is included in MultiSelectOptions) or msNone (in other case). The default value for MultiSelect property is always restored when ClearSelection is called. When dgSelectRow option is included in Options only the msRows multiselection is available in the grid. When dgMultiSelect option is included in Options the user's mouse and keyboard actions can change the current value of MultiSelect.

When no any row/column/cell is multiselected (MultiSelected = False) the multiselection is starting:

- When the user clicks on indicator (or on fixed data cell), the msRows multiselection is starting.
- When the user clicks on title cell and move mouse pointer down, the msCols multiselection is starting.
- When the user clicks on non-fixed data cell, the msCells multiselection is starting.
- When the user clicks on any data cell and the Ctrl key is hold down, the msRows multiselection is starting.
- When the user clicks on any data cell and the Shift key is hold down, the msCols multiselection is starting.

When the multiselection is just started (MultiSelected = True), the user can select additional row/col/cell:

- When the user clicks on any data cell and the Ctrl key is hold down, the additional single row/column is selected.
- When the user clicks on any data cell and the Shift key is hold down, all the next rows/columns/cells are selected.

Note. When msFixed option is included in MultiSelectOptions the fixed data cells (See also: FixedCols) are multiselected in the same way as non-fixed data cells.

The user can also start (and continue) multiselection with using keyboard:

- When the Shift key is hold down and the user press Up/Down arrow key (also PgUp/PgDn), the msRows multiselection is performed. When the Shift key is still hold down and the user press now Left/Right arrow key (also Home/End), the multiselection is changed to msCells.
- When the Shift key is hold down and the user press Left/Right arrow key (also Home/End), the msCols multiselection is performed. When the Shift key is still hold down and the user press now Up/Down arrow key (also PgUp/PgDn), the multiselection is changed to msCells.

Note. When not all of [msRows, msCols, msCells] options are included in MultiSelectOptions property some of the above actions may not work or may work in other way.

See also: TXCustomDBGrid.MultiSelectOptions, TXCustomDBGrid.MultiSelected, TXCustomDBGrid.ClearSelection

TXCustomDBGrid.MultiSelected

Specifies whenever any row, column or cell is multiselected.

```
property MultiSelected: Boolean; {* ver. 4.3 *}
```

Description
Read MultiSelected to determine is any row, column or cell currently multiselected in the grid. MultiSelected returns True when RowsSelected or ColsSelected returns True, otherwise MultiSelected return False. Use MultiSelected property to recognize is the grid currently in "multiselected mode".

MultiSelected is a read-only property.

See also: TXCustomDBGrid.MultiSelect, TXCustomDBGrid.MultiSelectOptions, TXCustomDBGrid.ColsSelected, TXCustomDBGrid.RowsSelected

TXCustomDBGrid.MultiSelectOptions

Specifies kinds of multiselection available in XDBGrid.

```
type
  TMultiSelectOption = msRows .. msFixed; // For C++Builder/Delphi 5, 6, 7
  TMultiSelectOption = msNone .. msFixed; // For C++Builder/Delphi 8 or higher
    (due problem with subset property) {* ver. 5.0 *}
  TMultiSelectOptions = set of TMultiSelectOption;
property MultiSelectOptions: TMultiSelectOptions; {* ver. 4.3 *}
```

Description
Set MultiSelectOptions to determind kinds of multiselection available in XDBGrid. Include dgMultiSelect option to Options property to activate MultiSelectOptions for user and developer. Include dgInternalSelect option to Options property to activate MultiSelectOptions for developer only. MultiSelectOptions is a set drawn from the following values:

Value	Meaning
msRows	Multiselection of rows is available.
msCols	Multiselection of columns is available.
msCells	Multiselection of cells (area) is available.
msFixed	Fixed data cells participate in multiselection. See also FixedCols.

See also: TXCustomDBGrid.MultiSelect, TXCustomDBGrid.MultiSelected

TXCustomDBGrid.Options

Specifies various display and behavioral properties of the XDBGrid.

```
type
  TXDBGridOption = (dgEditing, dgAlwaysShowEditor, dgTitles, dgIndicator,
    dgColumnResize, dgColLines, dgRowLines, dgTabs, dgRowSelect,
    dgAlwaysShowSelection, dgConfirmDelete, dgCancelOnExit, dgMultiSelect,
    dgExtendedSelect, dgInternalSelect,  dgRowResize, dgRowScroll,
    dgHotButtons, dgTitleButtons, dgTitleHeaders, dgTitleWidthOff,
    dgAllowDeleteOff, dgAllowInsertOff, dgAutoUnselectOff {obsolete},
    dgIndicatorMarkOff, dgMarkerAutoAlign, dgMarkerAutoSwitch,
    dgMarkerAutoToggle, dgMarkerAscendOnly, dgForceSequence, dgThumbTracking,
    dgRightMoving {obsolete});
  TXDBGridOptions = set of TXDBGridOption;
property Options: TXDBGridOptions;
```

Description

Set Options to include the desired properties for the XDBGrid component. Options is a set drawn from the following values:

Value	Meaning
dgEditing	The user can edit data using the grid. dgEditing is ignored if Options includes dgRowSelect.
dgAlwaysShowEditor	The grid is always in edit mode. That is, the user does not have to press Enter or F2 before editing the contents of a cell. dgAlwaysShowEditor does nothing unless dgEditing is also included in Options. dgAlwaysShowEditor is ignored if Options includes dgRowSelect. Other restrictions has been removed in version 5.3. {* ver. 5.3 *}
dgTitles	Titles appear at the top of the columns in the grid.
dgIndicator	A small pointer appears in the first column to indicate which row is current.
dgColumnResize	Columns can be resized or moved. See also ResizeOptions.
dgColLins	Lines appear between columns in the grid.
dgRowLines	Lines appear between the rows of the grid.
dgTabs	The user can navigate through the grid using the Tab and Shift+Tab keys.
dgRowSelect	The user can select an entire row, as well as selecting individual cells. If Options includes dgRowSelect, dgEditing and dgAlwaysShowEditor are ignored.
dgAlwaysShowSelection	The selected cell displays the focus rectangle even when the grid does not have focus.
dgConfirmDelete	A message box appears, asking for confirmation, when the user types Ctrl+Delete to delete a row in the grid.
dgCancelOnExit	When the user exits the grid from an inserted record to which the user made no modifications, the inserted record is not posted to the dataset. This prevents the inadvertent posting of empty records.
dgMultiSelect	More than one row/col/cell in the grid can be selected at a time. See also MultiSelectOptions. {* ver. 4.3 *}

dgExtendedSelect	If Options includes dgExtendedSelect, the user can take advantage extended selection. This option handles the following keyboard and mouse events: Ctrl+A (SelectAll), Ctrl+C or either Ctrl+Insert (CopyToClipboard) Shift+PgUp, Shift+PgDn, Shift+Home, Shift+End, Shift+Ctrl+Home, Shift+Ctrl+End, Alt+Shift+Click (SelectAll), Alt+Ctrl+Click (InvertAll), Alt+Click (UnselectAll). The dgExtendedSelect option is respected common with dgMultiSelect option only.
dgInternalSelect	If Options includes dgInternalSelect and excludes dgMultiSelect, the user can't change selection using standard keyboard or mouse actions, but the developer can use selection in run-time code.
dgRowResize	The user can resize height of rows using left mouse button.
dgRowScroll	If Options includes dgRowSelect, the user can scroll column as well using keyboard.
dgHotButtons	Highlights the title buttons as the mouse passes over them.
dgTitleButtons	Title of column can be used as button.
dgTitleHeaders	Headers appear at the top of the titles. Headers are visible only, if Options includes dgTitles.
dgTitleWidthOff	Width of title's caption is excluded from calculation of default width for column.
dgAllowDeleteOff	When the user types Ctrl+Delete to delete a row in the grid, action is ignored.
dgAllowInsertOff	When the user types Insert or Ctrl+Insert to insert new row, action is ignored.
dgAutoUnselectOff	When user clicks on the row, selected rows are not unselected. This option is used only when RowsSelection is True. {* ver. 6.0 *}
dgIndicatorMarkOff	If Options includes dgIndicator, the small pointer not appears in the indicator.
dgMarkerAutoAlign	Sorting markers drawn in the titles are aligned to the right side of title's caption.
dgMarkerAutoSwitch	When new sorting marker is set, all others markers are clear.
dgMarkerAutoToggle	When the user clicks on title, sorting marker is self toggled.
dgMarkerAscendOnly	When sorting marker is toggled, only ascending order is accepted.
dgForceSequence	This option forces sequence numbers for database records linked to the grid. This allow to realize proportional scrolling and AutoNumber for any dataset. See also: ForcedSequence, UpdateSequence, RecNumber, RecCount.
dgThumbTracking	The grid image updates while the user is dragging the thumb tab of the scrollbar. If dgThumbTracking is not included, the image does not update until the user releases the thumb tab in a new position.
dgRightMoving	Columns can be moved by using right (instead left) mouse button. This option is now obsolete and it can't be included in .NET version.

See also: TXCustomDBGrid.HintOptions, TXCustomDBGrid.OptionsEx, TXCustomDBGrid.OptionsExt, TXCustomDBGrid.OrderFields

TXCustomDBGrid.OptionsEx

Specifies another various display and behavioral properties of the XDBGrid.

```
type
  TXDBGridOptionEx = (dgBlankRow, dgTotalHeader, dgTotalValues, dgTotalFields,
    dgTotalFooter, dgTotalColLines, dgTotalHeaderBox, dgTotalValuesBox,
    dgTotalFieldsBox, dgTotalFooterBox, dgTotalDataBoxOnly,
    dgLoadCurrentOrder, dgAutoUpdateOrder, dgAutoAscendOrder,
    dgAutoKeepSelection, dgAutoUpdateSequence, dgDelayUpdateSequence,
    dgAutoUpdateTotals, dgDelayUpdateTotals, dgCalcInvisibleRows,
    dgCalcWholeDataSet, dgCalcSelectedRows, dgDelaySelectedRows,
    dgTotalHeaderSelected, dgTotalValuesSelected, dgTotalFooterSelected,
    dgSelectedAutoHidden, dgClickSelectOff);
  TXDBGridOptionsEx = set of TXDBGridOptionEx;
property OptionsEx: TXDBGridOptionsEx;
```

Description
Set OptionsEx to include the desired properties for the XDBGrid component. OptionsEx is a set drawn from the following values:

Value	Meaning
dgBlankRow	Blank row appears at the bottom of the columns in the grid.
dgTotalHeader	TotalHeader single row appears at the bottom of the grid.
dgTotalValues	TotalValues rows appear at the bottom of the grid.
dgTotalFields	TotalFields rows appear at the bottom of the grid.
dgTotalFooter	TotalFooter single row appears at the bottom of the grid.
dgTotalColLines	Lines appear between columns in the total rows (dgColLines must be included in Options). When IsGridThemed this option is ignored.
dgTotalHeaderBox	Box appears in the TotalHeader cells when TotalBox is tbAuto.
dgTotalValuesBox	Box appears in the TotalValues cells when TotalBox is tbAuto.
dgTotalFieldsBox	Box appears in the TotalFields cells when TotalBox is tbAuto.
dgTotalFooterBox	Box appears in the TotalFooter cells when TotalBox is tbAuto.
dgTotalDataBoxOnly	Box appear in the total cells only when total cell is not empty.
dgLoadCurrentOrder	Automatically SetupCurrentOrder from DataSet linked to the grid when DataSet is opened.
dgAutoUpdateOrder	Automatically ChangeDataSetOrder in DataSet linked to the grid when OrderFields changed or when DataSet is opened and dgLoadCurrentOrder in not included in OptionsEx property. See also OnOrderUpdated event.
dgAutoAscendOrder	When ChangeDataSetOrder method is performed automatically modify dgMarkerAscendOrder option in Options property depend on possibilities of current DataSet linked to the grid. See also RegisterChangeOrder procedure.
dgAutoKeepSelection	When RefreshDataSet or ChangeDataSetOrder method is performed the grid automatically saves and restores current state of selection. This option works effecively only when KeyFields property is defined. {* ver. 5.32 *}
dgAutoUpdateSequence	When dgForceSequence is included in Options property the grid automatically perform UpdateSequence when data in the grid has changed and sequence number must be recalculated. See also OnSequenceUpdated event.

dgDelayUpdateSequence	When dgAutoUpdateSequnece is included in OptionsEx property the grid delays call of UpdateSequnece to reduce number of calls. See also DelayInterval global variable. This option is not respected in design-time. {* ver. 5.0 *}
dgAutoUpdateTotals	Automatically perform UpdateTotals when data in the grid has changed or when SelectedRows changed and calculated values in total cells (TotalResult = trCalcValue) must be recalculated. See also OnTotalsUpdated event.
dgDelayUpdateTotals	When dgAutoUpdateTotals in included in OptionsEx property the grid delays call of UpdateTotals to reduce number of calls. See also DelayInterval global variable. This option is not respected in design-time. {* ver. 5.0 *}
dgCalcInvisibleRows	Perform CalcValue function for total cells also when total row is currently not visible in the grid.
dgCalcWholeDataSet	Apply UpdateTotals for all records in DataSet when Totals.Selected is False.
dgCalcSelectedRows	Apply UpdateTotals for SelectedRows or/and SelectedCols only when Totals.Selected is True.
dgDelaySelectedRows	Delays call of SelectedRowsChanged method to reduce number of calls. See also DelayInterval global variable, OnSelectedRowsChanged and OnSelectedColsChanged events. This option is not respected in design-time. {* ver. 5.0 *}
dgTotalHeaderSelected	Apply TotalHeader row to calculate total values for SelectedRows only.
dgTotalValuesSelected	Apply TotalValues rows to calculate total values for SelectedRows only.
dgTotalFooterSelected	Apply TotalFooter row to calculate total values for SelectedRows only.
dgSelectedAutoHidden	Automatically hide "Selected" total rows when there is no selected rows/columns in the grid (Totals.Selected is False). This option works only in run-time.
dgClickSelectOff	When dgMultiSelect and dgExtendedSelect are included in Options property simple click does'nt select a record. When this option is included in OptionsEx use Ctrl-Click to select single record. Using this option allows to permanently include dgMultiSelect option and select records only when the user really need this.
dgShowEditorByCharOff	When this option is included in OptionsEx and dgEditing is included in Options the character keys (like 1, 2, ..., A, B, ...) do not show inplace editor. You can start editing by Enter or F2 key only. {* ver. 6.00 *}

See also: TXCustomDBGrid.Options, TXCustomDBGrid.OptionsExt

TXCustomDBGrid.OptionsExt

Specifies another various display and behavioral properties of the XDBGrid.

```
type
  TXDBGridOptionExt = (dgAutoNumberColumn, dgAutoSelectColumn,
    dgAutoExpandColumn, dgAutoUpdateListItems, dgDelayUpdateListItems,
    dgGridStyleLists, dgMouseScrolling, dgMouseTracking, dgMarkerClearAllowed,
    dgScrollGridMode, dgScrollPageMode, dgForceSystemTheme,
    dgForceSystemButton, dgForceSystemMarker, dgForceCustomized,
    dgScaleByParentFont, dgIncrementalList); {* ver. 6.0 *}{* ver. 6.2 *}{*
    ver. 6.6 *}{* ver. 7.8 *}
  TXDBGridOptionsExt = set of TXDBGridOptionExt;
property OptionsExt: TXDBGridOptionsExt; {* ver. 4.3 *}
```

Description
Set OptionsExt to include the desired properties for the XDBGrid component. OptionsExt is a set drawn from the following values:

Value	Meaning
dgAutoNumberColumn	When dgAutoNumberColumn is included in OptionsExt property and Columns.State is csDefault the AddAutoNumber function is automatically performed when the dataset linked to the grid is opened. In design-time you must include this option before you click "Add All Fields" button in Column Editor. See also: DefAutoNumberTitle, DefAutoNumberWidth global variable.
dgAutoSelectColumn	When dgAutoSelectColumn is included in OptionsExt property and Columns.State is csDefault the AddAutoSelect function is automatically performed when the dataset linked to the grid is opened. In design-time you must include this option before you click "Add All Fields" button in Column Editor. See also: DefAutoSelectTitle, DefAutoSelectWidth global variable.
dgAutoExpandColumn	When dgAutoExpandColumn is included in OptionsExt property and Columns.State is csDefault the AddAutoExpand function is automatically performed when the dataset linked to the grid is opened. In design-time you must include this option before you click "Add All Fields" button in Column Editor. See also: DefAutoExpandTitle, DefAutoExpandWidth global variable. {* ver. 6.6 *}
dgAutoUpdateListItems	When poAutoLoadList option is included in PickOptions property, the grid automatically perform UpdateListItems when data in the grid has changed and auto-loaded PickList must be updated. See also OnListItemsUpdated event
dgDelayUpdateListItems	When dgAutoUpdateListItems is included in OptionsExt property, the grid delays call of UpdateListItems to reduce number of calls. See also DelayInterval global variable. This option is not respected in design-time. {* ver. 5.0 *}
dgGridStyleLists	When dgGridStyleLists is included in OptionsExt property, the PickList/PickText/LookupList is displayed with using new grid style. This new style allows also to use additional functionallity for lookup list defined as new options in ListOptions property. See also: ListDropDownDelay, ListMouseTracking global variable.
dgMouseScrolling	When the grid is not in edit mode and when you click left mouse button on the data cell and you still hold left mouse button pressed, you can move the mouse pointer in any direction to scroll visibled data in the grid. When dgMultiSelect option is included in Options and you move the mouse pointer, the selection will be changed according to mouse pointer.

dgMouseTracking	When the grid is not in edit mode and when you move the mouse pointer over the data cells, the focused cell will be changed according to mouse pointer.
dgMarkerClearAlowed	When the user clicks on a title with holding Ctrl key, the first sorting marker is cleared (none sorting order is allowed). {* ver. 5.1 *}
dgScrollGridMode	When the grid is vertically scrolled by using keyboard, scrollbar or mouse wheel all data rows are scrolled and current row in the grid leave unchanged. This option works only when dgForcedSequence is included in Options. {* ver. 5.31 *}
dgScrollPageMode	When the grid is vertically scrolled by using scrollbar's thumb button or mouse wheel and dgScrollGridMode option works effectively the data rows are scrolled page by page. {* ver. 5.31 *}
dgScrollRowsMode	When the grid is vertically scrolled by using mouse wheel all data rows are scrolled and current row in the grid leave unchanged instead of scrolling current row in the grid. This option works only when dgScrollGridMode are not included in OptionsExt. {* ver. 5.6 *}
dgForceSystemTheme	When FixedTheme is ftDefault you can replace default theme by system theme on Windows Vista and Windows 7. System theme is always used as default on Windows 8. {* ver. 5.32 *}
dgForceSystemButton	When ThemesEnabled or StylesEnabled is True you can replace default button in title cell by system button. System button is always used when SystemTheme is used. {* ver. 6.0 *}
dgForceSystemMarker	When ThemesEnabled or StylesEnabled is True you can replace default sorting marker in title cell by system marker. System marker is always used when SystemTheme is used. {* ver. 6.0 *}
dgForceCustomized	When Columns.State is csDefault (columns were not added to the grid by Columns Editor in design-time) you may force Columns.State = csCustomized after DataSet linked to the grid is first time open. This option causes that columns are not destroyed and re-created when DataSet linked to the grid is closed and re-opened to refresh data. To can change DataSet linked to the grid, you must call Columns.Clear first. {* ver. 5.32 *}
dgScaleByParentFont	When ParentFont is True and the parent font size is changed the width of grid columns is changed proportionally. {* ver. 6.2 *}
dgIncrementalList	When dgIncrementalList option is included in OptionsExt property Incremental Search function is active for each data and pick list in the grid. See also loIncrementalList option in ListOptions property and foIncrementalList in FilterGrid.Options. {* ver. 7.8 *}

See also: TXCustomDBGrid.Options, TXCustomDBGrid.OptionsEx

TXCustomDBGrid.OrderFields

Specifies one or more fields separated by comma that indicates current order in dataset linked to the grid.

```
property OrderFields: string;
property OrderFields: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description

Set OrderFields to appear sorting Markers to the right of a title's caption. Each time when OrderFields property changes and dgAutoUpdateOrder option is inluded in OptionsEx the ChangeDataSetOrder universal method is called. When dgAutoUpdateOrder is not included in OptionsEx or when DataSet linked to the grid is not registered to automatically change order the OnOrderChanged event is triggered to change sorting order in DataSet.

See also: TXCustomDBGrid.DetailFields, TXCustomDBGrid.Options, TXCustomDBGrid.BeginOrderUpdate, TXCustomDBGrid.EndOrderUpdate, TXCustomDBGrid.SetupOrderFields, TXCustomDBGrid.OnOrderChanged, TXColumnTitle.OrderIndex

TXCustomDBGrid.PopupToolMenu

Determines using ToolMenu as PopupMenu in the grid.

```
property PopupToolMenu: Boolean; {* ver. 8.0 *}
```

Description

Set PopupToolMenu to True to use ToolMenu as PopupMenu in the grid.

See also: TXCustomDBGrid.FillerToolMenu, TXCustomDBGrid.ToolMenu

TXCustomDBGrid.Position

Indicates a position of the active record in the grid.

```
property Position: Integer;
```

Description
Examine Position to determine position of the active record in the grid. Active record is the current record within the internal record buffer maintained by the grid. When DataLink is active the Position return value of ActiveRecord, otherwise the Position return 0.

Set Position to value from range 0 to DataRowCount-1 to select as current one of visibled records in the grid.

See also: TXCustomDBGrid.GotoPosition, TXCustomDBGrid.DisablePosition, TXCustomDBGrid.EnablePosition, TXCustomDBGrid.PositionDisabled, TXCustomDBGrid.DataRowCount

TXCustomDBGrid.RecCount

Indicates the total number of records in the grid.

```
property RecCount: Integer;
```

Description
Examine RecordCount to determine the total number of records in the grid. Applications might use this property with RecNumber to iterate through all the records in the grid.

When ForcedSequence is True, RecCount return total number of records in the grid for any linked dataset even when filter is active. When ForcedSequence is False, RecCount return RecordCount value for linked dataset.

This property allow to realize proportional scrolling and AutoNumber for any dataset. See also when you must call yourself UpdateSequence to retain correct RecNumber and RecCount value.

RecCount is a read-only property.

See also: TXCustomDBGrid.ForcedSequence, TXCustomDBGrid.UpdateSequence, TXCustomDBGrid.RecNumber, TXCustomDBGrid.Options

TXCustomDBGrid.RecNumber

Indicates ordinal position of the active record in the grid.

```
property RecNumber: Integer;
```

Description
Examine RecNumber to determine ordinal position of the active record in the grid. Active record is the current record within the internal record buffer maintained by the grid. See also: ActiveRecord.

When ForcedSequence is True, RecNumber return ordinal position of the active record for any linked dataset even when filter is active. When ForcedSequence is False, RecNumber return RecNo value for linked dataset. This value indicates correct ordinal position in the grid only when IsSequenced is True.

Applications might use this property with RecCount to iterate through all the records in the grid. This property allow to realize proportional scrolling and AutoNumber for any dataset. See also when you must call yourself UpdateSequence to retain correct RecNumber value.

See also: TXCustomDBGrid.ForcedSequence, TXCustomDBGrid.UpdateSequence, TXCustomDBGrid.RecCount, TXCustomDBGrid.Options, TXColumn.AutoNumber

TXCustomDBGrid.ResizedColumn

Specifies the lately resized column in the grid.

```
property ResizedColumn: TXColumn;
```

Description
Read ResizedColumn in OnColumnResize event handler to determine the lately resized column in the grid. The value of this property is valid only in OnColumnResize event. The value of ResizedColumn can be nil, when more than one column have been changed, a column moved or width of Indicator changed.

ResizedColumn is a read-only property.

See also: TXCustomDBGrid.ResizeOptions, TXCustomDBGrid.OnColumnResize

TXCustomDBGrid.ResizeOptions

Specifies various columns resize properties of the XDBGrid.

```
type
  TXResizeOption = (roColumnResize, roOptimalWidth, roDefaultWidth,
    roColumnMoving);
  TXResizeOptions = set of TXResizeOption;
property ResizeOptions: TXResizeOptions;
```

Description
Set ResizeOptions to include various columns resize properties for the XDBGrid. Include dgColumnResize option to Options property to activate ResizeOptions. ResizeOptions is a set drawn from the following values:

Value	Meaning
roColumnResize	Columns can be resized.
roOptimalWidth	Double click can set optimal width for the column. See also OptimalWidth.
roDefaultWidth	Middle mouse button click can set default width for the column. See also DefaultWidth.
roColumnMoving	Columns can be moved. See also dgRightMoving option in Options.

See also: TXCustomDBGrid.Options, TXColumn.OptimalWidth, TColumn.DefaultWidth, TXCustomDBGrid.OnColumnMoving, TCustomDBGrid.OnColumnMoved

TXCustomDBGrid.RowHeight

Specifies the height (in pixels) of all data rows.

```
property RowHeight: Integer;
```

Description
RowHeight is calculated as DefaultRowHeight + RowLineWidth.

RowHeight is a read-only property.

See also: TXCustomDBGrid.TitleHeight, TXCustomDBGrid.ScrollBarHeight, TXCustomDBGrid.RowLineHeight

TXCustomDBGrid.RowLineWidth

Specifies the width (in pixels) of the lines that separate the rows of the grid.

```
property RowLineWidth: Integer;
```

Description

If the Options property include dgRowLines, RowLineWidth is 1, otherwise RowLineWidth is 0.

RowLineWidth is a read-only property.

See also: TXCustomDBGrid.ColLineWidth, TXCustomDBGrid.TitleHeight, TXCustomDBGrid.ScrollBarHeight, TXCustomDBGrid.RowHeight

TXCustomDBGrid.RowsChanged

Indicates the changes were made in SelectedRows list.

```
property RowsChanged: Boolean; {* ver. 4.3 *}
```

Description

The RowsChanged flag indicates the changes were made in SelectedRows list after BeginSelect method was called at the start of the changes. When the changes were made the RowsChanged is True. The RowsChanged flag is cleared by EndSelect method.

RowsChanged is a read-only property.

See also: TXCustomDBGrid.BeginSelect, TXCustomDBGrid.EndSelect, TXCustomDBGrid.ColsChanged, TXCustomDBGrid.OnSelectedRowsChanged

TXCustomDBGrid.RowsSelected

Specifies whenever any row or data cell is multiselected.

```
property RowsSelected: Boolean; {* ver. 4.3 *}
```

Description

Read RowsSelected to determine is any row or data cell currently multiselected in the grid. RowsSelected returns True when RowsSelection is True and SelectedRows list is not empty, otherwise RowsSelected return False.

RowsSelected is a read-only property.

See also: TXCustomDBGrid.MultiSelected, TXCustomDBGrid.ColsSelected, TXCustomDBGrid.RowsSelection, TXCustomDBGrid.SelectedRows

TXCustomDBGrid.RowsSelection

Specifies whenever any row or data cell is multiselected.

```
property RowsSelection: Boolean; {* ver. 4.3 *}
```

Description
Read RowsSelection to determine is multiselection of rows or cells currently active. RowsSelection returns True when MultiSelect is msRows or msCells, otherwise RowsSelection return False. When RowsSelection is True, the SelectedRows list contains bookmarks of selected rows (msRows) or bookmarks of selected area (msCells).

RowsSelection is a read-only property.

See also: TXCustomDBGrid.ColsSelection, TXCustomDBGrid.MultiSelect, TXCustomDBGrid.SelectedRows, TXCustomDBGrid.RowsSelected

TXCustomDBGrid.ScrollBarHeight

Specifies the height (in pixels) of horizontal scroll bar.

```
property ScrollBarHeight: Integer;
```

Description
ScrollBarHeight is depend on system configuration settings. When horizontal scroll bar is invisible, ScrollBarHeight is 0.

ScrollBarHeight is a read-only property.

See also: TXCustomDBGrid.ScrollBarWidth, TXCustomDBGrid.ScrollBars, TXCustomDBGrid.TitleHeight, TXCustomDBGrid.RowHeight, TXCustomDBGrid.RowLineHeight

TXCustomDBGrid.ScrollBars

Specifies whether the grid includes horizontal and vertical scroll bars.

```
property ScrollBars: TScrollStyle;
```

Description
See TCustomGrid.ScrollBars to read more. See also AutoHidden.

See also: TXCustomDBGrid.ScrollBarHeight, TXCustomDBGrid.ScrollBarWidth, TXCustomDBGrid.ScrollProp

TXCustomDBGrid.ScrollBarWidth

Specifies the width (in pixels) of vertical scroll bar.

```
property ScrollBarWidth: Integer;
```

Description
ScrollBarWidth is depend on system configuration settings. When vertical scroll bar is invisible, ScrollBarWidth is 0.

ScrollBarWidth is a read-only property.

See also: TXCustomDBGrid.ScrollBarHeight, TXCustomDBGrid.ScrollBars, TXCustomDBGrid.ColLineWidth, TXCustomDBGrid.IndicatorWidth

TXCustomDBGrid.ScrollGridMode

Determines whether the grid scrolls all data rows at once.

```
property ScrollGridMode: Boolean; {* ver. 5.31 *}
```

Description
Use ScrollGridMode to determine whether the grid scroll all data rows at once instead of scrolling current row. The ScrollGridMode return True, when dgScrollGridMode option is included in OptionsExt property and dgForceSequence option is included in Options property.

ScrollGridMode is a read-only property.

See also: TXCustomDBGrid.ScrollPageMode, TXCustomDBGrid.OptionsExt

TXCustomDBGrid.ScrollPageMode

Determines whether the grid scrolls data rows page by page.

```
property ScrollPageMode: Boolean; {* ver. 5.31 *}
```

Description
Use ScrollPageMode to determine whether the grid scroll data rows page by page. The ScrollPageMode return True, when dgScrollPageMode option is included in OptionsExt property and ScrollGridMode property is True.

ScrollPageMode is a read-only property.

See also: TXCustomDBGrid.ScrollGridMode, TXCustomDBGrid.OptionsExt

TXCustomDBGrid.ScrollRowsMode

Determines whether the mouse wheel scrolls all data rows instead of scrolling current row.

```
property ScrollRowsMode: Boolean; {* ver. 5.6 *}
```

Description
Use ScrollRowsMode to determine whether the mouse wheel scrolls all data rows instead of scrolling current row. The ScrollRowsMode return True, when dgScrollRowsMode option is included in OptionsExt property and dgScrollGridMode option is not included in OptionsExt property.

ScrollRowsMode is a read-only property.

See also: TXCustomDBGrid.ScrollGridMode, TXCustomDBGrid.ScrollPageMode

TXCustomDBGrid.ScrollProp

Indicates the TXScrollProp obect that determines additional possibilities for scrollbars.

```
property ScrollProp: TXScrollProp;
```

Description
The ScrollProp property points to TXScrollProp object that determines additional possibilities for scrollbars. The ScrollProp possibilities are practicable only when FlatSBMode is accessible.

See also: TXCustomDBGrid.ScrollBars, TXCustomDBGrid.FlatSBMode, TXScrollProp

TXCustomDBGrid.Search

Indicates the TXDBGridSearch that represents search settings for the grid.


```
property Search: TXDBGridSearch; {* ver. 6.2 *}
```

Description
The Search property points to TXDBGridSearch object that determines search settings for the grid.

See also: TXDBGridSearch

TXCustomDBGrid.SearchPanelHeight

Specifies the height (in pixels) of search panel.

```
property SearchPanelHeight: Integer; {* ver. 6.2 *}
```

Description
If the search panel is visible, SearchPanelHeight specifies the height of the panel in pixels, otherwise SearchPanelHeight is 0.

SearchPanelHeight is not a read-only property, but you should not self change this property in code.

See also: TXCustomDBGrid.SearchPanelVisible, TXCustomDBGrid.SearchPanelFocused

---

TXCustomDBGrid.SearchPanelFocused

Specifies the search panel is focused in the grid.

```
property SearchPanelFocused: Boolean; {* ver. 7.3 *}
```

Description
If the search panel is focused in the grid the SearchPanelFocused is True, otherwise SearchPanelFocused is False.

SearchPanelFocused is a read-only property.

See also: TXCustomDBGrid.ShowSearchPanel, TXCustomDBGrid.CloseSearchPanel, TXCustomDBGrid.SearchPanelHeight, TXCustomDBGrid.SearchPanelVisible

---

TXCustomDBGrid.SearchPanelVisible

Specifies the search panel is visible in the grid.

```
property SearchPanelVisible: Boolean; {* ver. 6.2 *}
```

Description
If the search panel is visible in the grid the SearchPanelVisible is True, otherwise SearchPanelVisible is False.

SearchPanelVisible is a read-only property.

See also: TXCustomDBGrid.ShowSearchPanel, TXCustomDBGrid.CloseSearchPanel, TXCustomDBGrid.SearchPanelHeight, TXCustomDBGrid.SearchPanelFocused

TXCustomDBGrid.SelectCellColor

Specifies the alternate highlight color to highlight focused cell in the grid.

```
property SelectCellColor: TColor;
```

Description
Set SelectCellColor to alternate highlight color to highlight focused cell in the grid. SelectCellColor is also respected for whole current row when dgRowSelect option is included in Options.

The SelectCellColor property can handle two special colors: clDefault, clNone.
Select clDefault value to use DefaultSelectCellColor value (clHighlight).
Select clNone value when the focused cell should not be highlighted.

See also: TXCustomDBGrid.SelectionColor, TXCustomDBGrid.SelectRowColor, TXCustomDBGrid.StripeColor, TXCustomDBGrid.EditorColor, TXCustomDBGrid.FocusRect

TXCustomDBGrid.SelectedCols

Specifies a set of columns that are selected in the grid.

```
property SelectedCols: TXColumnList; {* ver. 4.3 *}
```

Description
Use the properties and methods of the TXColumnList object returned by SelectedCols to

- Determine the number of columns in the grid that are selected.
- Determine whether the current column in the grid is selected.
- Determine whether a particular column in the grid is selected.

SelectedCols are highlighted in the grid only when dgMultiSelect or either dgInternalSelect option is included in Options.

See also: TXColumnList, TXCustomDBGrid.Options, TXCustomDBGrid.SelectedRows

TXCustomDBGrid.SelectedColumn

Specifies the column component for the currently selected cell in the grid.

```
property SelectedColumn: TXColumn; {* ver. 4.3 *}
```

Description
Set SelectedColumn to move focus to a particular column in the grid. Read SelectedColumn to obtain access to the column component for the currently selected cell. If there is not currently selected cell, SelectedColumn is nil. For example, when an entire row is selected, SelectedColumn is nil.

See also: TXCustomDBGrid.Columns, TXCustomDBGrid.SelectedField, TXCustomDBGrid.SelectedIndex

TXCustomDBGrid.SelectedField

Specifies the field component for the currently selected cell in the grid.

```
property SelectedField: TField;
```

Description
Set SelectedField to move focus to a particular field in the grid. Read SelectedField to obtain access to the field component for the currently selected cell. If there is not currently selected cell, SelectedField is nil. For example, when an entire row is selected, SelectedField is nil.

See also: TCustomDBGrid.Fields, TXCustomDBGrid.SelectedColumn, TXCustomDBGrid.SelectedIndex

TXCustomDBGrid.SelectedIndex

Specifies the index of the currently selected column in the Columns array.

```
property SelectedIndex: Integer;
```

Description
Set SelectedIndex to move focus to a column in the grid that is identified by position. Read SelectedIndex to determine which column in the grid has focus. A value of 0 indicates the first data column, 1 is the second data column, and so on. SelectedIndex is -1 if there is no currently selected column.

If the Options property includes dgIndicator, the index of the data column given by SelectedIndex will differ from the index of the physical column in the grid.

To access the field component for the selected column, use the SelectedField property.

To access the column component for the selected column, use the SelectedColumn property.

See also: TXCustomDBGrid.Columns, TXCustomDBGrid.SelectedColumn, TXCustomDBGrid.SelectedField

TXCustomDBGrid.SelectedRows

Specifies a set of bookmarks for all the records in the dataset that correspond to rows selected in the grid.

```
property SelectedRows: TXBookmarkList;
```

Description
Use the properties and methods of the TXBookmarkList object returned by SelectedRows to

- Determine the number of rows in the grid that are selected.
- Determine whether the current record in the dataset is selected.
- Determine whether a particular record in the dataset is selected.
- Delete all selected rows from the dataset.

SelectedRows are highlighted in the grid only when dgMultiSelect or either dgInternalSelect option is included in Options.

See also: TXBookmarkList, TXCustomDBGrid.Options, TXCustomDBGrid.SelectedCols

TXCustomDBGrid.SelectionAnchor

Indicates first row in the grid, that will be using during multirow selection.

```
const
  EmptyBookmark = '';
  EmptyBookmark = nil; // Delphi 2009, 2010, XE
property SelectionAnchor: TBookmarkStr;
property SelectionAnchor: TBookmark; // Delphi 2009, 2010, XE
```

Description
Set SelectionAnchor to the dataset's Bookmark to select first row using during multirow selection. Last row using during multirow selection is always current row.

Set EmptyBookmark as SelectionAnchor to clear selection anchor. Selection anchor is visible, if dgIndicator option is included in Options.

See also: TXCustomDBGrid.SelectRows, TXCustomDBGrid.InvertRows, TXCustomDBGrid.UnselectRows, TXCustomDBGrid.ClearSelection

TXCustomDBGrid.SelectionColor

Specifies the color for multiselected rows in the grid.

```
property SelectionColor: TColor;
```

Description
Set SelectionColor to change color of multiselected rows in the grid. SelectionColor is respected only when dgMultiSelect or dgInternalSelect option is included in Options.

The SelectionColor property can handle two special colors: clDefault, clNone.
Select clDefault value to use DefaultSelectionColor value (clGradientActiveCaption).
Select clNone value to leave unchanged color for multiselected rows (clHighlight).

See also: TXCustomDBGrid.SelectCellColor, TXCustomDBGrid.SelectRowColor, TXCustomDBGrid.StripeColor, TXCustomDBGrid.EditorColor

TXCustomDBGrid.SelectRowColor

Specifies the alternate highlight color to highlight current row in the grid.

```
property SelectRowColor: TColor;
```

Description
Set SelectRowColor to alternate highlight color to highlight current row in the grid. The selected color should be alternate to SelectCellColor, which is used as background color of focused cell. SelectRowColor is respected only when dgRowSelect option is not included in Options.

The SelectRowColor property can handle two special colors: clDefault, clNone.
Select clDefault value to use DefaultSelectRowColor value (clGradientInactiveCaption).
Select clNone value when the current row should not be highlighted.

See also: TXCustomDBGrid.SelectCellColor, TXCustomDBGrid.SelectionColor, TXCustomDBGrid.StripeColor, TXCustomDBGrid.EditorColor

TXCustomDBGrid.Settings

Represents columns layout management for TXDBGrid.

```
property Settings: TXDBGridSettings;
```

Description
The Settings property points to TXDBGridSettings object that holds additional properties for columns layout management. They allow to save/load columns layout (width, order, visibility, etc.) to/from registry or *.ini file.

See also: TXDBGridSettings

TXCustomDBGrid.StretchMode

Indicates whether the grid should change width of columns so that thay exactly fit the width of the control.

```
property StretchMode: Boolean;
```

Description
Set StretchMode to True to cause the columns exactly fit the width of grid. Each column's width is changed proportional. StretchMode respects individual settings of WidthMin/WidthMax properties for any column and global settings of StretchWidthMin/StretchWidthMax properties for all columns in the grid.

Set StretchMode to False to return to previous width of columns.

See also: TXCustomDBGrid.StretchGrid, TXCustomDBGrid.StretchWidthMax, TXCustomDBGrid.StretchWidthMin

TXCustomDBGrid.StretchWidthMax

Specifies the maximum width for any column when stretch mode is on.

```
property StretchWidthMax: Integer;
```

Description
Set StretchWidthMax to the maximum value the Width property for any column can take. The StretchWidthMin and StretchWidthMax properties define the total range over which Width property can't vary when StretchMode is True.

See also: TXCustomDBGrid.StretchMode, TXCustomDBGrid.StretchGrid, TXCustomDBGrid.StretchWidthMin

TXCustomDBGrid.StretchWidthMin

Specifies the minimum width for any column when stretch mode is on.

```
property StretchWidthMin: Integer;
```

Description
Set StretchWidthMin to the minimum value the Width property for any column can take. The StretchWidthMin and StretchWidthMax properties define the total range over which Width property can't vary when StretchMode is True.

See also: TXCustomDBGrid.StretchMode, TXCustomDBGrid.StretchGrid, TXCustomDBGrid.StretchWidthMax

TXCustomDBGrid.StripeColor

Specifies the alternate color for the grid to draw horizontal striped cells.

```
property StripeColor: TColor;
```

Description
Set StripeColor to alternate color to draw horizontal striped cells in the grid. The selected color should be alternate to main Color value, which is always used as color of second stripe.

The StripeColor property can handle two special colors: clDefault, clNone.
Select clDefault value to use DefaultStripeColor value (clBtnHighlight).
Select clNone value when the grid should not be striped.

See also: TXCustomDBGrid.SelectCellColor, TXCustomDBGrid.SelectionColor, TXCustomDBGrid.SelectRowColor, TXCustomDBGrid.EditorColor

TXCustomDBGrid.StyleElements

Specifies the style elements that are used by the grid.

```
type
  TStyleElements = set of (seFont, seClient, seBorder);
property StyleElements: TStyleElements; {* ver. 5.3 *} // For
C++Builder/Delphi XE2 or higher
```

Description
Use StyleElements to customize the style of the grid. If a custom style is enabled, you can enable or disable some style elements in the grid. StyleElements is a set of values that specify which elements of the current style are to be applied to this control. By default, all the elements of the style are enabled. These are the possible values of StyleElements:

Value	Meaning
seFont	The grid uses the font color defined in the style (for data cells).
seClient	The grid uses the background color defined in the style (for data cells).
seBorder	The grid uses the border and scroll bars of the style.

Notice. The fixed and selected cells are always styled when custom style is enabled.

See also: StylesEnabled, TXCustomDBGrid.SystemStyleElements

TXCustomDBGrid.SystemStyleElements

Specifies the system style elements that are used by the grid.

```
type
  TSystemStyleElements = set of (sseFixed, sseText, sseButton, sseMarker,
    sseData, sseFont, sseClient);
property SystemStyleElements: TSystemStyleElements; {* ver. 5.3 *} // For
C++Builder/Delphi XE2 or higher
```

Description
Use SystemStyleElements to customize the system style of the grid. Starting from Windows Vista, if a system style is enabled, you can enable or disable some style elements in the grid (only when DrawingStyle is gdsThemed). SystemStyleElements is a set of values that specify which elements of the system style are to be applied to this control. By default, most elements of the style are enabled. These are the possible values of SystemStyleElements:

Value	Meaning
sseFixed	The grid uses the background image defined in the system style (for fixed cells).
sseText	The grid uses the text color defined in the system style (for fixed cells).
sseButton	The grid uses the expand button defined in the system style (for title cells).
sseMarker	The grid uses the sorting marker defined in the system style (for title cells).
sseData	The grid uses the background image defined in the system style (for selected cells).
sseFont	The grid uses the font color defined in the style (for data cells).
sseClient	The grid uses the background color defined in the style (for data cells).

Notice. By default, the fixed and selected cells are styled when system style is enabled (starting from Windows Vista). You may exclude all (or some) elements from SystemStyleElements to go back to an earlier themed style of the grid. See also IsSystemStyle function.

See also: TXCustomDBGrid.IsSystemStyle, StylesEnabled, TXCustomDBGrid.StyleElements

TXCustomDBGrid.TitleHeight

Specifies the height (in pixels) of title's row.

```
property TitleHeight: Integer;
```

Description
If the Options property include dgTitles, TitleHeight is calculated as height of first row + RowLineWidth, otherwise TitleHeight is 0. If any Expandable column is Expanded, TitleHeight is calculated as sum of rows height occupied by titles.

TitleHeight is a read-only property.

See also: TXCustomDBGrid.ScrollBarHeight, TXCustomDBGrid.RowHeight, TXCustomDBGrid.BlankHeight, TXCustomDBGrid.TotalHeight, TXCustomDBGrid.DataRowsHeight, TXCustomDBGrid.DataHeight

TXCustomDBGrid.TitleImages

Determines which image list is associated with the titles.

```
property TitleImages: TImageList;
```

Description
Use TitleImages to provide a customized list of bitmaps that can be displayed to the left of a title's caption. Individual columns specify the image from this list that should appear by setting their ImageIndex property.

See also: TXCustomDBGrid.IndicatorImages, TXCustomDBGrid.TitleMarkers, TXColumn.Images, TXColumnTitle.ImageIndex

TXCustomDBGrid.TitleLinesCount

Specifies number of lines for multi-line titles of columns in the grid.

```
property TitleLinesCount: Integer;
```

Description
Set TitleLinesCount to specify number of lines for multi-line titles of columns in the grid. TitleLinesCount is only meaningful if the Options property includes dgTitles.

See also: TXCustomDBGrid.LinesCount, TXCustomDBGrid.HeaderLinesCount

TXCustomDBGrid.TitleMarkers

Specifies custom images that are drawn as sorting markers in the titles.

```
property TitleMarkers: TImageList;
```

Description
Use TitleMarkers to provide a customized list of bitmaps that can be displayed to the right of a title's caption as sorting markers. Individual columns specify the image from this list that should appear by setting their MarkerIndex property.

See also: TXCustomDBGrid.Markers, TXCustomDBGrid.TitleImages

TXCustomDBGrid.TitleRowCount

Specifies the number of all title rows.

```
property TitleRowCount: Integer;
```

Description
If the Options property include dgTitles, TitleRowCount specifies count of rows occupied by titles, otherwise TitleRowCount is 0. TitleRowCount may be greater then 1 if any Expandable column is Expanded.

TotalRowCount is a read-only property.

See also: TXCustomDBGrid.DataRowCount, TXCustomDBGrid.TotalRowCount

TXCustomDBGrid.ToolMenu

Indicates the ready tool menu to associate with TXDBGrid.

```
property ToolMenu: TPopupMenu; {* ver. 8.0 *}
```

Description
The ToolMenu property points to internal predefined auto-created TXDBGridToolMenu object that can be associated with any PopupMenu in TXDBGrid. ToolMenu is customized by properties ToolMenuItems, ToolMenuSubItems, ToolMenuSubItemsEx.

See also: TXCustomDBGrid.FillerToolMenu, TXCustomDBGrid.PopupToolMenu

TXCustomDBGrid.ToolMenuItems

Specifies the items that are visible in tool menu.

```
type
  TXDBGridToolMenuItem = set of (tmOpen, tmRefresh, tmShowColumns, tmColumns,
    tmShowAutoFilter, tmAutoFilter, tmShowMultiSelect, tmMultiSelect,
    tmSelectedRow, tmStripedRows, tmTreeView, tmSettings, tmClose);
  TXDBGridToolMenuItems = set of TXDBGridToolMenuItem ;
property ToolMenuItems: TXDBGridToolMenuItems; {* ver. 8.0 *}
```

Description
Use ToolMenuItems to customize items of ToolMenu. By default, all main items are visible in ToolMenu. To create short ToolMenu you shoud check tmShow... items and uncheck next items for submenu. You should also uncheck all items that user should not use in that grid. After the ToolMenu is created, your selection is additinally validated to hide redundant items (eg. tmTreeView).

Value	Meaning
tmOpen	Perform DataSet.Open method.
tmRefresh	Perform RefreshDataSet method.
tmShowColumns	Show XDBColumnsDialog dialog.
tmColumns	Popup submenu Columns.
tmShowAutoFilter	Switch AutoFilter property in FiterGrid object.
tmAutoFilter	Popup submenu AutoFilter.
tmShowMultiSelect	Switch dgMultiSelect option in Options property.
tmMultiSelect	Popup submenu MultiSelect.
tmSelectedRow	Switch gdoSelectedRow option in DrawingOptions property.
tmStripedRows	Switch gdoStripedRows option in DrawingOptions property.
tmTreeView	Switch AutoActive property in TreeView object.
tmSettings	Popup submenu Settings.
tmClose	Perform DataSet.Close method.

See also: TXCustomDBGrid.ToolMenu, TXCustomDBGrid.ToolMenuSubItems, TXCustomDBGrid.ToolMenuSubItemsEx

TXCustomDBGrid.ToolMenuSubItems

Specifies the items that are visible in tool submenu.

```
type
  TXDBGridToolMenuSubItem = set of (tmColumnsShow, tmColumnsStretch,
    tmColumnsDefaultWidth, tmColumnsOptimalWidth, tmColumnsIncludeTitles,
    tmColumnsIncludeButtons, tmColumnsLoad, tmColumnsSave,
    tmColumnsLoadFromFile, tmColumnsSaveToFile, tmAutoFilterActive,
    tmAutoFilterFiltered, tmAutoFilterClear, tmAutoFilterLoad,
    tmAutoFilterSave, tmAutoFilterLoadFromFile, tmAutoFilterSaveToFile,
    tmMultiSelectActive, tmMultiSelectRows, tmMultiSelectCols,
    tmMultiSelectCells, tmMultiSelectFixed);
  TXDBGridToolMenuSubItems = set of TXDBGridToolMenuSubItem ;
property ToolMenuSubItems: TXDBGridToolMenuSubItems; {* ver. 8.0 *}
```

Description

Use ToolMenuSubItems to customize subitems of ToolMenu. By default, all main subitems are visible in ToolMenu. You should also uncheck all subitems that user should not use in that grid. After the ToolMenu is created, your selection is additinally validated to hide redundant subitems (eg. tmMultiSelectFixed).

Value	Meaning
tmColumnsShow	Show XDBColumnsDialog dialog.
tmColumnsStretch	Switch StretchMode property.
tmColumnsDefaultWidth	Perform AdjustColumnsWidth(False) method.
tmColumnsOptimalWidth	Perform AdjustColumnsWidth(True, Buttons) method.
tmColumnsIncludeTitles	Switch dgTitleWidthOff option in Options property.
tmColumnsIncludeButtons	Switch Buttons for AdjustColumnsWidth(True, Buttons) method.
tmColumnsLoad	Load Columns.Settttings from internal ColumnSettings property.
tmColumnsSave	Save Columns.Settings to internal ColumnSettings property.
tmColumnsLoadFromFile	Load Columns.Settttings from column/text file (*.clm, *.txt).
tmColumnsSaveToFile	Save Columns.Settings to column/text file (*.clm, *.txt).
tmAutoFilterActive	Switch FilterGrid.AutoFilter property.
tmAutoFilterFiltered	Switch FilterGrid.Filtered property.
tmAutoFilterClear	Perform FilterGrid.ClearAutoFilter method.
tmAutoFilterLoad	Load FilterGrid.Settttings from internal FilterSettings property.
tmAutoFilterSave	Save FilterGrid.Settttings to internal FilterSettings property.
tmAutoFilterLoadFromFile	Load FilterGrid.Settttings from filter/text file (*.flt, *.txt).
tmAutoFilterSaveToFile	Save FilterGrid.Settttings to filter/text file (*.flt, *.txt).
tmMultiSelectActive	Switch dgMultiSelect option in Options property.
tmMultiSelectRows	Switch msRows option in MultiSelectOptions property.
tmMultiSelectCols	Switch msCols option in MultiSelectOptions property.
tmMultiSelectCells	Switch msCells option in MultiSelectOptions property.
tmMultiSelectFixed	Switch msFixed option in MultiSelectOptions property.

See also: TXCustomDBGrid.ToolMenu, TXCustomDBGrid.ToolMenuSubItems, TXCustomDBGrid.ToolMenuSubItemsEx

TXCustomDBGrid.ToolMenuSubItemsEx

Specifies the items that are visible in tool submenu.

```
type
  TXDBGridToolMenuSubItemEx = set of (tmSettingsLoadLayout,
    tmSettingsSaveLayout, tmSettingsLoad, tmSettingsSave,
    tmSettingsEraseFromFile, tmSettingsLoadFromFile, tmSettingsSaveToFile,
    tmSettingsEraseFromRegistry, tmSettingsLoadFromRegistry,
    tmSettingsSaveToRegistry);
  TXDBGridToolMenuSubItemsEx = set of TXDBGridToolSubMenuItemEx ;
property ToolMenuSubItemsEx: TXDBGridToolMenuSubItemsEx; {* ver. 8.0 *}
```

Description
Use ToolMenuSubItemsEx to customize subitems of ToolMenu. By default, all main subitems are visible in ToolMenu. You should also uncheck all subitems that user should not use in that grid. After the ToolMenu is created, your selection is additinally validated to hide redundant subitems (eg. tmSettingsLoadLayout/tmSettingsSaveLayout).

Value	Meaning
tmSettingsLoadLayout	Switch soLoadLayout option in Settings.Options property.
tmSettingsSaveLayout	Switch soSaveLayout option in Settings.Options property.
tmSettingsLoad	Load Settttings.Layout from internal LayoutSettings property.
tmSettingsSave	Save Settttings.Layout to internal LayoutSettings property.
tmSettingsEraseFromFile	Erase Settttings.Layout from settings file (*.ini).
tmSettingsLoadFromFile	Load Settttings.Layout from settings/text file (*.ini, *.txt).
tmSettingsSaveToFile	Save Settttings.Layout to settings/text file (*.ini, *.txt).
tmSettingsEraseFromRegistry	Erase Settttings.Layout from registry.
tmSettingsLoadFromRegistry	Load Settttings.Layout from registry.
tmSettingsSaveToRegistry	Save Settttings.Layout to registry.

See also: TXCustomDBGrid.ToolMenu, TXCustomDBGrid.ToolMenuSubItems, TXCustomDBGrid.ToolMenuSubItemsEx

TXCustomDBGrid.TotalFieldsRowCount

Specifies the number of TotalFields rows.

```
property TotalFieldsRowCount: Integer;
```

Description
If the OptionsEx property include dgTotalFields, TotalFieldsRowCount specifies count of all TotalFields rows, otherwise TotalFieldsRowCount is 0. TotalFieldsRowsCount depend on number of records in Totals.DataSource.DataSet and value of Totals.DataMaxRow property.

TotalFieldsRowCount is a read-only property.

See also: TXCustomDBGrid.TotalFooterRowCount, TXCustomDBGrid.TotalHeaderRowCount, TXCustomDBGrid.TotalValuesRowCount, TXCustomDBGrid.TotalRowCount

TXCustomDBGrid.TotalFooterRowCount

Specifies the number of TotalFooter's rows.

```
property TotalFooterRowCount: Integer;
```

Description
If the OptionsEx property include dgTotalFooter, TotalFooterRowCount is 1, otherwise TotalFooterRowCount is 0.

TotalFooterRowCount is a read-only property.

See also: TXCustomDBGrid.TotalFieldsRowCount, TXCustomDBGrid.TotalHeaderRowCount, TXCustomDBGrid.TotalValuesRowCount, TXCustomDBGrid.TotalRowCount

TXCustomDBGrid.TotalHeaderRowCount

Specifies the number of TotalHeader's rows.

```
property TotalHeaderRowCount: Integer;
```

Description
If the OptionsEx property include dgTotalHeader, TotalHeaderRowCount is 1, otherwise TotalHeaderRowCount is 0.

TotalHeaderRowCount is a read-only property.

See also: TXCustomDBGrid.TotalFooterRowCount, TXCustomDBGrid.TotalFieldsRowCount, TXCustomDBGrid.TotalValuesRowCount, TXCustomDBGrid.TotalRowCount

TXCustomDBGrid.TotalHeight

Specifies the height (in pixels) of all total rows.

```
property TotalHeight: Integer;
```

Description
If the OptionsEx property include dgTotalHeader, dgTotalValues, dgTotalFields or dgTotalFooter, TotalHeight is calculated as sum of height of all total rows, otherwise TotalHeight is 0.

TotalHeight is a read-only property.

See also: TXCustomDBGrid.TitleHeight, TXCustomDBGrid.BlankHeight, TXCustomDBGrid.DataRowsHeight, TXCustomDBGrid.DataHeight

TXCustomDBGrid.TotalRowCount

Specifies the number of all total rows.

```
property TotalRowCount: Integer;
```

Description
If the OptionsEx property include dgTotalHeader, dgTotalValues, dgTotalFields or dgTotalFooter, TotalRowCount specifies count of all total rows, otherwise TotalRowCount is 0.

TotalRowCount is a read-only property.

See also: TXCustomDBGrid.DataRowCount, TXCustomDBGrid.TitleRowCount

TXCustomDBGrid.Totals

Indicates the TXDBGridTotals object that determines common properties for all total cells.

```
property Totals: TXDBGridTotals;
```

Description
The Totals property points to TXDBGridTotals object that holds common properties for all total cells.

See also: TXDBGridTotals

TXCustomDBGrid.TotalValuesRowCount

Specifies the number of TotalValues rows.

```
property TotalValuesRowCount: Integer;
```

Description
If the OptionsEx property include dgTotalValues, TotalValuesRowCount specifies count of all TotalValues rows, otherwise TotalValuesRowCount is 0. TotalValuesRowsCount depend on maximum value of TotalValues.Count property calculated for all columns defined in the grid.

TotalValuesRowCount is a read-only property.

See also: TXCustomDBGrid.TotalFooterRowCount, TXCustomDBGrid.TotalHeaderRowCount, TXCustomDBGrid.TotalFieldsRowCount, TXCustomDBGrid.TotalRowCount

TXCustomDBGrid.TreeView

Indicates the TXDBGridTreeView that represents tree view for the grid.

```
property TreeView: TXDBGridTreeView; {* ver. 7.0 *}
```

Description
The TreeView property points to TXDBGridTreeView object that determines tree view for the grid.

See also: TXDBGridTreeView

TXCustomDBGrid.UpdateInProgress

Indicates the update mode is in progres in the grid.

```
property UpdateInProgress: Boolean; {* ver. 4.3 *}
```

Description
Use UpdateInProgress to determine the grid is currently in update mode.

UpdateInProgress is True, when one of InUpdateListItems, InUpdateSequence or InUpdateTotals is True.

UpdateInProgress is a read-only property.

See also: TXCustomDBGrid.InUpdateListItems, TXCustomDBGrid.InUpdateSequence, TXCustomDBGrid.InUpdateTotals

TXCustomDBGrid.WheelScrollRows

Specifies the number of rows that are scrolled for each notch that the mouse wheel is rotated.

```
property WheelScrollRows: Integer;
```

Description
Set WheelScrollRows to specify number of rows to scroll when one mouse wheel notch is rotated. When WheelScrollRows is 0 the system value of WheelScrollLines will be used. WheelScrollLines can be set under Mouse in the Windows Control Panel. When WheelScrollRows is -1 the number of visible rows in the grid will be used (one page will be scrolled). To get the true number of rows that are scrolled for each notch that the mouse wheel is rotated use MouseWheelScrollRows function.

Using mouse wheel with Ctrl key pressed allows to scroll the grid always by 1 row. Using mouse wheel with Shift key pressed allows to scroll the grid always by 1 page. {* ver. 5.6 *}

See also: TXCustomDBGrid.MouseWheelScrollRows

TXCustomDBGrid.AdjustColumnsWidth

Adjusts the width of the grid columns to the size of the data visible in the grid.

```
procedure AdjustColumnsWidth(OptimalWidth, ButtonPlace: Boolean); {* ver. 7.5 *}
```

Description
Call AdjustColumnsWidth to adjust width of all columns to the size of data visible in the grid.

When parameter OptimalWidth is True this method calls function OptimalWidth. When parameter OptimalWidth is False this method calls function DefaultWidth. When parameter ButtonPlace is True the result of new column width is increased of button width (useful for editor button in data cell or filter button in title cell).

See also: TXColumn.OptimalWidth

TXCustomDBGrid.BeginOrderUpdate

Increments internal OrderChangeLock property when the data order in the grid changes.

```
procedure BeginOrderUpdate;
```

Description
XDBGrid calls BeginOrderUpdate internally before making changes that affect the data order in the grid. Once the change is complete, the XDBGrid calls EndOrderUpdate, which decrements internal OrderChangeLock property.

While OrderChangeLock is greater than 0, the grid does not repaint its cells and does not call OnOrderChanged event. BeginOrderUpdate and EndOrderUpdate prevent the grid from partially order changes.

See also: TXCustomDBGrid.EndOrderUpdate, TXCustomDBGrid.OrderFields

TXCustomDBGrid.BeginSelect

Increments internal SelectLock property when selected rows or columns changes.

```
procedure BeginSelect;
```

Description
XDBGrid calls BeginSelect internally before making group of changes in SelectedRows and/or SelectedCols. Once the change is complete, the XDBGrid calls EndSelect, which decrements internal SelectLock property.

When SelectLock is decremented to 0, EndSelect calls OnSelectedRowsChanged and/or OnSelectedColsChanged events. The OnSelectedRowsChanged event is triggered only when RowsChanged flag is True. The OnSelectedColsChanged event is triggered only when ColsChanged flag is True. It allows to call each event once - after group of changes was made in SelectedRows/SelectedCols. You can examine RowsChanged/ColsChanged flags in your event handlers.

See also: TXCustomDBGrid.EndSelect, TXCustomDBGrid.OnSelectedRowsChanged, TXCustomDBGrid.OnSelectedColsChanged

TXCustomDBGrid.CancelSelect

This method is now obsolete.

```
procedure CancelSelect; { obsolete }
```

Description
XDBGrid calls CancelSelect internally when no changes are made in SelectedRows/SelectedCols after BeginSelect was performed. This method is now obsolete and it calls EndSelect. {* ver. 4.3 *}

See also: TXCustomDBGrid.EndSelect, TXCustomDBGrid.BeginSelect

TXCustomDBGrid.CellExpandPoint

Returns a point suitable to open expand form for the column.

```
function CellExpandPoint(Column: TXColumn; AWidth: Integer): TPoint; {* ver. 6.6 *}
```

Description
Call CellExpandPoint to determine a point suitable to open expand form for the Column. AWidth parameter specifies a width of expanded form. AWidth parameter is needed only when BiDiMode is bdRightToLeft. The result of function is returned as global screen coordinates.

See also: TXCustomDBGrid.OnCellExpand

TXCustomDBGrid.ChangeDataSetFilter

Changes filter properties in dataset linked to the grid.

```
procedure ChangeDataSetFilter(AFiltered: Boolean; AFilter: string;
AOnFilterRecord: TFilterRecordEvent; AKeyFields: string = ''); virtual; {*
ver. 6.0 *}
```

Description
Use ChangeDataSetFilter to change filter properties in DataSet linked to the grid. AFiltered parameter contains new state of filtering, AFilter parameter describes new filter condition and AOnFilterRecord parameter contains new OnFilterRecord event handler. AKeyFields parameter specifies the field or fields which allows to identify records in dataset. When the parameter is omitted the current value of KeyFields property will be used to identify records in DataSet.

Usually, you not need to call this method directly. To change any filter property in DataSet linked to the grid simply change the property Filter, Filtered or OnFilterRecord in TXDBGrid.

See also: TXCustomDBGrid.Filter, TXCustomDBGrid.Filtered, TXCustomDBGrid.OnFilterRecord, TXCustomDBGrid.UpdateFilter

TXCustomDBGrid.ChangeDataSetOrder

Changes sorting order in dataset linked to the grid.

```
function ChangeDataSetOrder(AOrderFields: string = ''): Boolean; virtual;
function ChangeDataSetOrder(AOrderFields: WideString = ''): Boolean; virtual;
// C++Builder/Delphi 2006, 2007 Win32 only
```

Description
Use ChangeDataSetOrder to change sorting order in DataSet linked to the grid. AOrderFields parameter describes new order in DataSet. When the parameter is omitted the current value of OrderFields property will be set as new order for DataSet.

When dgAutoUpdateOrder is included in OptionsEx this method is called automatically (without parameter) when OrderFields (sorting markers) changed. When dgAutoUpdateOrder is not included in OptionsEx, OnOrderChanged event is triggered. You can write this event handler and call method ChangeDataSetOrder with any value for AOrderFields parameter. It's especially useful, when you want to replace any calculated or lookup fields to it's equivalent in DataSet before you call this method.

This method is effective only for DataSet's registered by RegisterChangeOrder or RegisterCustomChangeOrder. The result of this function is True when DataSet linked to the grid is registered or False when DataSet is not registered. When sorting order was changed in DataSet OnOrderUpdated event is triggered.

ChangeDataSetOrder is a universal method for all registered DataSet's. All standard DataSet descendats (except unidirectional) are registered by default to automatically change order (ADO, BDE, CDS, DBX, IBX). This method calls internally ChangeIndexFields, ChangeOrderFields, ChangeSQLOrderFields or the custom method registered by RegisterCustomChangeOrder procedure depend on the ClassName of DataSet linked to the grid.

See also: SetupCurrentOrder, CurrentDataSetOrder.

See also: TXCustomDBGrid.ChangeIndexFields, TXCustomDBGrid.ChangeOrderFields, TXCustomDBGrid.ChangeSQLOrderFields, TXCustomDBGrid.UpdateOrder

TXCustomDBGrid.ChangeIndexFields

Changes sorting order in a table linked to the grid.

```
procedure ChangeIndexFields(AOrderFields: string = ''; PropName: string = '';
Suffix: string = '');
procedure ChangeIndexFields(AOrderFields: WideString = ''; PropName: string =
''; Suffix: string = ''); // C++Builder/Delphi 2006, 2007 Win32 only
```

Description
Use ChangeIndexFields (in OnOrderChanged event handler) to change sorting order in a table linked to the grid. AOrderFields parameter describes new order in DataSet. When the parameter is omitted the current value of OrderFields property will be set as new order for DataSet. AOrderFields parameter must be compatible with format of OrderFields property. It's internally converted to the format appropriate for IndexFieldNames property.

PropName parameter describes index fields published property in DataSet linked to the grid. When PropName parameter is omitted (or empty) the default published property called "IndexFieldNames" will be expected in DataSet to change index fields. For other names, you must explicity specify the name of the property in PropName parameter.

`{* ver. 4.31 *}` Suffix parameter describes index fields modifier. When Suffix parameter is omitted (or empty) the default suffix `'ASC;DESC;'` will be used to change current sorting order in data set. For other suffixes, you must explicity specify the ASC and DESC modifiers when they are accepted by DataSet. For example, the AnyDAC's data sets accept the modifiers `':A;:D;:N'`. Use semicolon to separate ASC and DESC modifiers. When none modifier is accepted by data set you must register the change order method with parameter AscOnly.

You can use this method only for DataSets which support IndexFieldNames as a method of specifying the index to use for a table. The sorting order will be changed only for indexed fields in the table. For other fields you will hear Beep only.

This method is internally registered for: TTable, TADOTable, TADODataSet, TIBTable, TIBClientDataSet, TClientDataSet, TSimpleDataSet, TSQLClientDataSet, TBDEClientDataSet.

See also: ChangeDataSetOrder, CurrentIndexFields.

See also: TXCustomDBGrid.ChangeDataSetOrder, TXCustomDBGrid.ChangeOrderFields, TXCustomDBGrid.ChangeSQLOrderFields, TXCustomDBGrid.UpdateOrder

TXCustomDBGrid.ChangeOrderFields

Changes sorting order in a dataset linked to the grid.

```
procedure ChangeOrderFields(AOrderFields: string = ''; PropName: string = '');
procedure ChangeOrderFields(AOrderFields: WideString = ''; PropName: string =
''); // C++Builder/Delphi 2006, 2007 Win32 only
```

Description
Use ChangeOrderFields (in OnOrderChanged event handler) to change sorting order in a dataset linked to the grid. AOrderFields parameter describes new order in DataSet. When the parameter is omitted the current value of OrderFields property will be set as new order for DataSet. AOrderFields parameter must be compatible with format of OrderFields property. It replaces fields in ORDER BY clause of CommandText property.

PropName parameter describes command text published property in DataSet linked to the grid. When PropName parameter is omitted (or empty) the default published property called "CommandText" will be expected in DataSet to change order fields. For other names, you must explicity specify the name of the property in PropName parameter.

You can use this method only when DataSet linked to the grid allow to specify SQL SELECT statement as String or WideString published property. The sorting order will be changed only when AOrderFields

parameter will be accepted in ORDER BY clause. In other case (lookup or calculated fields) you will hear Beep only.

This method is internally registered for: TADODataSet, TIBClientDataSet, TClientDataSet, TSimpleDataSet, TSQLClientDataSet, TBDEClientDataSet.

See also: ChangeDataSetOrder, CurrentOrderFields.

See also: TXCustomDBGrid.ChangeIndexFields, TXCustomDBGrid.ChangeDataSetOrder, TXCustomDBGrid.ChangeSQLOrderFields, TXCustomDBGrid.UpdateOrder

TXCustomDBGrid.ChangeSQLOrderFields

Changes sorting order in a query linked to the grid.

```
procedure ChangeSQLOrderFields(AOrderFields: string = ''; PropName: string = '');
procedure ChangeSQLOrderFields(AOrderFields: WideString = ''; PropName: string = ''); // C++Builder/Delphi 2006, 2007 Win32 only
```

Description
Use ChangeSQLOrderFields (in OnOrderChanged event handler) to change sorting order in a query linked to the grid. AOrderFields parameter describes new order in DataSet. When the parameter is omitted the current value of OrderFields property will be set as new order for DataSet. AOrderFields parameter must be compatible with format of OrderFields property. It replaces fields in ORDER BY clause of SQL property.

PropName parameter describes SQL published property in DataSet linked to the grid. When PropName parameter is omitted (or empty) the default published property called "SQL" will be expected in DataSet to change order fields. For other names, you must explicity specify the name of the property in PropName parameter.

You can use this method only when DataSet linked to the grid allow to specify SQL SELECT statement as TStrings or TWideStrings published property. The sorting order will be changed only when AOrderFields parameter will be accepted in ORDER BY clause. In other case (lookup or calculated fields) you will hear Beep only.

This method is internally registered for: TQuery, TADOQuery, TIBQuery, TIBDataSet.

See also: ChangeDataSetOrder, CurrentSQLOrderFields.

See also: TXCustomDBGrid.ChangeIndexFields, TXCustomDBGrid.ChangeOrderFields, TXCustomDBGrid.ChangeDataSetOrder, TXCustomDBGrid.UpdateOrder

TXCustomDBGrid.ClearSelection

Clears all multiselected records/columns/cells in the grid.

```
procedure ClearSelection;
```

Description
Use ClearSelection to unselect all records/columns/cells in the grid. ClearSelection restores also default value for MultiSelect property and clears SelectionAnchor.

Note. You sholud call this method especially after dataset linked to the grid was refreshed and dataset's bookmarks are no longer valid.

See also: TXCustomDBGrid.SelectionAnchor, TXCustomDBGrid.MultiSelect, TXCustomDBGrid.MultiSelectOptions, TXCustomDBGrid.SelectedRows, TXCustomDBGrid.SelectedCols

TXCustomDBGrid.CloseSearchPanel

Close the search panel visible in the grid.

```
procedure CloseSearchPanel; {* ver. 6.2 *}
```

Description
Use CloseSearchPanel to hide search panel when it's visible in the grid. The user can perform this action by Esc key.

See also: TXCustomDBGrid.ShowSearchPanel, TXCustomDBGrid.UpdateSearchPanel, TXCustomDBGrid.SearchPanelHeight, TXCustomDBGrid.SearchPanelVisible

TXCustomDBGrid.ColGetText

Performs additional processing for the Text retrieved by grid's Column.

```
type
  TTextKind = (tkEdit, tkDisplay, tkReport);
function ColGetText(Column: TXColumn; const Text: string; TextKind:
TTextKind): string; virtual;
```

Description
ColGetText method performs additional processing for a Text value, when the Text is retrieved from dataset. ColGetText is designed to format Text value depend on Column's properties. ColGetText considers PickText and PickOptions properties and finally calls OnColGetText event. The TextKind parameter determines text destination.

See also: TXCustomDBGrid.ColSetText, TXCustomDBGrid.OnColGetText, , TXCustomDBGrid.GetTotalText

TXCustomDBGrid.ColSetText

Performs additional processing for the Text stored by grid's Column.

```
function ColSetText(Column: TXColumn; const Text: string): string; virtual;
```

Description
ColSetText method performs additional processing for a Text value, when the Text is stored to dataset. ColSetText is designed to format Text value depend on Column's properties. ColSetText calls OnColSetText event and finally considers PickText and PickOptions properties.

See also: TXCustomDBGrid.ColGetText, TXCustomDBGrid.OnColSetText

TXCustomDBGrid.ColumnAtDepth

Returns the column object associated with parent column of the given column.

```
function ColumnAtDepth(Col: TXColumn; ADepth: Integer): TXColumn;
```

Description
ColumnAtDepth is used to calculate MasterCol used in the CalcTitleRect and FullTitleRect methods. ADepth parameter determines a depth in titles area starting from 0.

See also: TXColumn.Expandable, TXColumn.Expanded, TXColumn.ExpandCols

TXCustomDBGrid.ColumnByName

Finds a column based on a specified field name.

```
function ColumnByName(const FieldName: string; FromIndex: Integer = 0):
TXColumn;
function ColumnByName(const FieldName: WideString; FromIndex: Integer = 0):
TXColumn; // C++Builder/Delphi 2006, 2007 Win32 only
```

Description

Call ColumnByName to retrieve column information for a column when only the field's name is known. FieldName is the name of a field stored in FieldName property in any column. ColumnByName returns the TXColumn object for the specified field. If the field can not be found in any column, an exception is raised. Specify FromIndex to continue searching duplicated FieldName starting from specified column index.

See also: TXCustomDBGrid.FindColumn

TXCustomDBGrid.CompareOrderFields

Compares two OrderFields strings.

```
function CompareOrderFields(F1, F2: string): Boolean;
function CompareOrderFields(F1, F2: WideString): Boolean; // C++Builder/Delphi
2006, 2007 Win32 only
```

Description

Use CompareOrderFields to compare two OrderFields strings field by field without case sensitive. The result is True, if both values determine the same sorting order. Otherwise, the result is False. This function is called before ChangeDataSetOrder changes sorting order in DataSet linked to the grid.

See also: TXCustomDBGrid.ChangeDataSetOrder, TXCustomDBGrid.OrderFields

TXCustomDBGrid.CopyToClipboard

Copies the cells selected in the grid to the Clipboard.

```
procedure CopyToClipboard(AddCaptions: Boolean = False); {* ver. 4.3 *}{* ver.
6.7 *}
```

Description

Use CopyToClipboard to replace the contents of the Clipboard with the text of cells MultiSelected in the grid. CopyToClipboard does not clear the Clipboard if no cell is selected (MultiSelected = False). If no cell is selected, CopyToClipboard does nothing. Set AddCaptions to True to add title captions of selected columns. {* ver. 6.7 *}

If Options include dgExtendedSelect, the user can copy selected cells to Clipboard by using Ctrl+C or Ctrl+Shift+C {* ver. 6.7 *} key.

See also: TXCustomDBGrid.ForEachDataCell, TXCustomDBGrid.MultiSelected

TXCustomDBGrid.CurrentDataSetOrder

Retrieves current sorting order from dataset linked to the grid.

```
function CurrentDataSetOrder: string; virtual;
function CurrentDataSetOrder: WideString; virtual; // C++Builder/Delphi 2006, 2007 Win32 only
```

Description

Use CurrentDataSetOrder to retrieve sorting order from DataSet linked to the grid.

This method is effective only for DataSet's registered by RegisterChangeOrder or RegisterCustomChangeOrder. The result of this function is compatible with format of OrderFields property.

CurrentDataSetOrder is a universal method for all registered DataSet's. All standard DataSet descendats (except unidirectional) are registered by default to automatically change order (ADO, BDE, CDS, DBX, IBX). This method calls internally CurrentIndexFields, CurrentOrderFields, CurrentSQLOrderFields or the custom function registered by RegisterCustomChangeOrder procedure depend on the ClassName of DataSet linked to the grid.

See also: SetupCurrentOrder, ChangeDataSetOrder.

See also: TXCustomDBGrid.CurrentIndexFields, TXCustomDBGrid.CurrentOrderFields, TXCustomDBGrid.CurrentSQLOrderFields, TXCustomDBGrid.SetupCurrentOrder

TXCustomDBGrid.CurrentIndexFields

Retrieves current sorting order from a table linked to the grid.

```
function CurrentIndexFields(PropName: string = ''; Suffix: string = ''): string;
function CurrentIndexFields(PropName: string = ''; Suffix: string = ''): WideString; // C++Builder/Delphi 2006, 2007 Win32 only
```

Description

Use CurrentIndexFields to retrieve sorting order from a table linked to the grid. The result of this function is compatible with format of OrderFields property.

PropName parameter describes index fields published property in DataSet linked to the grid. When PropName parameter is omitted (or empty) the default published property called "IndexFieldNames" will be expected in DataSet to retrieve current sorting order. For other names, you must explicity specify the name of the property in PropName parameter.

`{* ver. 4.31 *}` Suffix parameter describes index fields modifier. When Suffix parameter is omitted (or empty) the default suffix `'ASC;DESC;'` will be accepted to retrieve current sorting order from data set. For other suffixes, you must explicity specify the ASC and DESC modifiers when they are accepted by DataSet. For example, the AnyDAC's data sets accept the modifiers `':A;:D;:N'`. Use semicolon to separate ASC and DESC modifiers. When none modifier is accepted by data set you must register the change order method with parameter AscOnly.

This method is internally registered for: TTable, TADOTable, TADODataSet, TIBTable, TIBClientDataSet, TClientDataSet, TSimpleDataSet, TSQLClientDataSet, TBDEClientDataSet.

See also: CurrentDataSetOrder, ChangeIndexFields.

See also: TXCustomDBGrid.CurrentDataSetOrder, TXCustomDBGrid.CurrentOrderFields, TXCustomDBGrid.CurrentSQLOrderFields, TXCustomDBGrid.SetupCurrentOrder

TXCustomDBGrid.CurrentOrderFields

Retrieves current sorting order from a dataset linked to the grid.

```
function CurrentOrderFields(PropName: string = ''): string;
function CurrentOrderFields(PropName: string = ''): WideString; //
C++Builder/Delphi 2006, 2007 Win32 only
```

Description
Use CurrentOrderFields to retrieve sorting order from dataset linked to the grid. The result of this function is compatible with format of OrderFields property.

PropName parameter describes command text published property in DataSet linked to the grid. When PropName parameter is omitted (or empty) the default published property called "CommandText" will be expected in DataSet to retrieve current sorting order. For other names, you must explicity specify the name of the property in PropName parameter.

This method is internally registered for: TADODataSet, TIBClientDataSet, TClientDataSet, TSimpleDataSet, TSQLClientDataSet, TBDEClientDataSet.

See also: CurrentDataSetOrder, ChangeOrderFields.

See also: TXCustomDBGrid.CurrentIndexFields, TXCustomDBGrid.CurrentSQLOrderFields, TXCustomDBGrid.SetupCurrentOrder

TXCustomDBGrid.CurrentSQLOrderFields

Retrieves current sorting order from a query linked to the grid.

```
function CurrentSQLOrderFields(PropName: string = ''): string;
function CurrentSQLOrderFields(PropName: string = ''): WideString; //
C++Builder/Delphi 2006, 2007 Win32 only
```

Description
Use CurrentSQLOrderFields to retrieve sorting order from a query linked to the grid. The result of this function is compatible with format of OrderFields property.

PropName parameter describes SQL published property in DataSet linked to the grid. When PropName parameter is omitted (or empty) the default published property called "SQL" will be expected in DataSet to retrieve current sorting order. For other names, you must explicity specify the name of the property in PropName parameter.

This method is internally registered for: TQuery, TADOQuery, TIBQuery, TIBDataSet.

See also: CurrentDataSetOrder, ChangeOrderFields.

See also: TXCustomDBGrid.CurrentIndexFields, TXCustomDBGrid.CurrentOrderFields, TXCustomDBGrid.CurrentDataSetOrder, TXCustomDBGrid.SetupCurrentOrder

TXCustomDBGrid.DefaultDataLineColor

Indicates default color of the data lines in the grid.

```
function DefaultDataLineColor: TColor; {* ver. 4.3 *}
```

Description
Call DefaultDataLineColor to get default color of data lines in the grid. DefaultDataLineColor is used when DataColLineColor or DataRowLineColorproperty is set to clDefault color.

See also: TXCustomDBGrid.DataColLineColor, TXCustomDBGrid.DataRowLineColor

TXCustomDBGrid.DefaultEditorColor

Indicates default color for grid's editor.

```
function DefaultEditorColor: TColor;
```

Description
Call DefaultEditorColor to get default color for grid's editor. DefaultEditorColor is used when EditorColor property is set to clDefault color.

See also: TXCustomDBGrid.EditorColor

TXCustomDBGrid.DefaultHighlightText

Indicates default color of text for focused or multiselected cells in the grid.

```
function DefaultHighlightText: TColor; {* ver. 4.3 *}
```

Description
Call DefaultHighlightText to get default color of text for focused or multiselected cells in the grid. DefaultHighlightText is used when HighlightText property is set to clDefault color.

See also: TXCustomDBGrid.HighlightText

TXCustomDBGrid.DefaultSelectCellColor

Indicates default color for focused cell in the grid.

```
function DefaultSelectCellColor: TColor;
```

Description
Call DefaultSelectCellColor to get default color for focused cell in the grid. DefaultSelectCellColor is used when SelectCellColor property is set to clDefault color.

See also: TXCustomDBGrid.SelectCellColor

TXCustomDBGrid.DefaultSelectionColor

Indicates default color for multiselected rows in the grid.

```
function DefaultSelectionColor: TColor;
```

Description
Call DefaultSelectionColor to get default color for multiselected rows in the grid. DefaultSelectionColor is used when SelectionColor property is set to clDefault color.

See also: TXCustomDBGrid.SelectionColor

TXCustomDBGrid.DefaultSelectRowColor

Indicates default color for current row in the grid.

```
function DefaultSelectRowColor: TColor;
```

Description
Call DefaultSelectRowColor to get default color for current row in the grid. DefaultSelectRowColor is used when SelectRowColor property is set to clDefault color.

See also: TXCustomDBGrid.SelectRowColor

TXCustomDBGrid.DefaultStripeColor

Indicates default color for striped rows in the grid.

```
function DefaultStripeColor: TColor;
```

Description
Call DefaultStripeColor to get default color for striped rows in the grid. DefaultStripeColor is used when StripeColor property is set to clDefault color.

See also: TXCustomDBGrid.StripeColor

TXCustomDBGrid.DisableGrid

Disables response on events incoming to XDBGrid component.

```
procedure DisableGrid;
```

Description
Call DisableGrid prior to iterating through a large number of records in the dataset connected to XDBGrid to prevent grid from updating every time the active record changes. Disabling grid prevents flicker and speeds performance because data does not need to be written to the display and the user can't change iterating process.

DisableGrid calls internally DisableControls and sets Enabled property to False.

DisableGrid is especially useful for Print and Preview method which iterates through the dataset connected to XDBGrid.

See also: TXCustomDBGrid.GridDisabled, TXCustomDBGrid.EnableGrid

TXCustomDBGrid.DisablePosition

Stores position in the grid and disables data display in data-aware controls associated with grid's dataset.

```
procedure DisablePosition;
```

Description
Call DisablePosition prior to iterating through a large number of records in the dataset connected to XDBGrid to prevent grid from updating every time the active record changes. Disabling grid prevents flicker and speeds performance because data does not need to be written to the display and the user can't change iterating process.

DisablePosition stores internally current row's position in the grid and a Bookmark to the active record in grid's dataset, then calls DisableControls.

Current row's position and active record can be restored by EnablePosition.

See also: TXCustomDBGrid.PositionDisabled, TXCustomDBGrid.EnablePosition

TXCustomDBGrid.DropDownForm

Drops down the form for the column.

```
function DropDownForm(Column: TXColumn; Title: Boolean): Boolean; {* ver. 7.1 *}
```

Description
Call DropDownForm to drop down the form for the Column. As now the DropDownForm method is used to drop down FilterForm only. This function returns True, if list was shown, otherwise it returns False.

See also: TXCustomDBGrid.DropDownList, TXCustomDBGrid.DropDownRect

TXCustomDBGrid.DropDownList

Drops down the list for the column.

```
function DropDownList(Column: TXColumn; Title: Boolean): Boolean;
```

Description
Call DropDownList to drop down the list for the Column. As now the DropDownList method is used to drop down FilterList only. This function returns True, if list was shown, otherwise it returns False.

See also: TXCustomDBGrid.DropDownForm, TXCustomDBGrid.DropDownRect

TXCustomDBGrid.DropDownMenu

Drops down the PopupMenu for the column.

```
function DropDownMenu(Column: TXColumn; Title: Boolean; PopupMenu: TPopupMenu = nil): Boolean;
```

Description
Call DropDownMenu to drop down the PopupMenu for the Column in DropDownPoint. If PopupMenu parameter is nil the default PopupMenu for the Column will be drop down. If Title parameter is True the default PopupMenu is Column.Title.DropDownMenu. If Title parameter is False the default PopupMenu is Column.DropDownMenu. If Column parameter is nil the default PopupMenu is FillerPopupMenu (Title=True) or either IndicatorPopupMenu (Title=False). This function returns True, if PopupMenu was shown, otherwise it returns False.

See also: TXCustomDBGrid.DropDownPoint, TXColumn.DropDownMenu, TXColumnTitle.DropDownMenu

TXCustomDBGrid.DropDownPoint

Returns a point suitable to drop down menu for the column.

```
function DropDownPoint(Column: TXColumn; Title: Boolean): TPoint;
```

Description
Call DropDownPoint to determine a point suitable to drop down menu for the Column. If Title parameter is True the function returns left lower corner of the Column title. If Title parameter is False the function returns left lower corner of data cell in current row. If Column parameter is nil the function returns a point for the filler (Title=True) or either for the indicator (Title=False). This function is internally used by DropDownMenu method. The result of function is returned as global screen coordinates.

See also: TXCustomDBGrid.DropDownMenu

TXCustomDBGrid.DropDownRect

Returns a rectangle suitable to drop down list for the column.

```
function DropDownRect(Column: TXColumn; Title: Boolean): TRect;
```

Description
Call DropDownRect to determine a rectangle suitable to drop down list for the Column. If Title parameter is True the function returns a rectangle of the Column title. If Title parameter is False the function returns a rectangle of data cell in current row. If Column parameter is nil the function returns a rectangle for the filler (Title=True) or either for the indicator (Title=False). This function is internally used by DropDownList method. The result of function is returned as global screen coordinates.

See also: TXCustomDBGrid.DropDownList

TXCustomDBGrid.EnableGrid

Re-enables response on events incoming to XDBGrid component.

```
procedure EnableGrid;
```

Description
Call EnableGrid to permit response on events incoming to XDBGrid component after a prior call to DisableGrid.

See also: TXCustomDBGrid.GridDisabled, TXCustomDBGrid.DisableGrid

TXCustomDBGrid.EnablePosition

Restores position in the grid and enables data display in data-aware controls associated with grid's dataset.

```
procedure EnablePosition(UseCurrentRec: Boolean = False);
```

Description
Call EnablePosition to permit response on events incoming to XDBGrid component after a prior call to DisablePosition. EnablePosition restores last raw position in the grid (and last active record in grid's dataset), then call EnableControls.

If UseCurrentRec is False, EnablePosition restores active record used before DisablePosition. Last active record can be properly restored, if internally stored Bookmark is still valid. Otherwise, current record in grid's dataset is unchanged.

If UseCurrentRec is True, EnablePosition restores only last row position in the grid and leaves current record in grid's dataset unchanged. In this way, you may select any new current record in dataset before you call EnablePosition.

See also: TXCustomDBGrid.PositionDisabled, TXCustomDBGrid.DisablePosition

TXCustomDBGrid.EndOrderUpdate

Decrements internal OrderChangeLock property.

```
procedure EndOrderUpdate;
```

Description
XDBGrid calls EndOrderUpdate internally after making changes that affect the data order in the grid. EndOrderUpdate returns internal OrderChangeLock property to the state it was in before the BeginOrderUpdate method was called at the start of the changes.

While OrderChangeLock is greater than 0, the grid does not repaint its cells and does not call OnOrderChanged event. When OrderChangeLock returns to 0, painting is reenabled and OnOrderChanged event is fired. BeginOrderUpdate and EndOrderUpdate prevent the grid from partially order changes.

See also: TXCustomDBGrid.BeginOrderUpdate, TXCustomDBGrid.OrderFields

TXCustomDBGrid.EndSelect

Decrements internal SelectLock property.

```
procedure EndSelect;
```

Description
XDBGrid calls EndSelect internally after making group of changes in SelectedRows and/or SelectedCols list. EndSelect decrements internal SelectLock property to return to the state it was before the BeginSelect method was called at the start of the changes.

When SelectLock is decremented to 0, EndSelect calls OnSelectedRowsChanged, OnSelectedColsChanged and OnSelectionChanged events. The OnSelectedRowsChanged event is triggered only when RowsChanged flag is True. The OnSelectedColsChanged event is triggered only when ColsChanged flag is True. The OnSelectionChanged event is triggered when RowsChanged or ColsChanged flag is True. It allows to call each event once - after group of changes was made in SelectedRows/SelectedCols list. You can examine RowsChanged/ColsChanged flag in your event handlers.

Alter all EndSelect clears RowsChanged and ColsChanged flags.

See also: TXCustomDBGrid.BeginSelect, TXCustomDBGrid.OnSelectedRowsChanged, TXCustomDBGrid.OnSelectedColsChanged

TXCustomDBGrid.FilterRecord

Filters records according to values in each FilterList.

```
procedure FilterRecord(DataSet: TDataSet; var Accept: Boolean); {* ver. 6.0 *}
```

Description
FilterRecord is called each time a different record in the dataset becomes the current record and internal filtering is active in the grid (when FilterGrid.Active is True). Accept parameter returns True when current record contains values from each not empty FilterList. When current record is internally accepted by the grid and Filtered property is True and OnFilterRecord is assigned, the OnFilterRecord event handler is also called to return finally result in Accept parameter.

When internal filtering is not active (FilterGrid.AutoUpdate is False), you can call this method from TDataSet.OnFilterRecord event handler, to check the current record contains values suitable to each FilterList.

See also: TXCustomDBGrid.OnFilterRecord

TXCustomDBGrid.FilterUpdated

Calls OnFilterUpdated event handler.

```
procedure FilterUpdated; {* ver. 6.0 *}
```

Description
FilterUpdated method is called by the grid when FilterList or FilterText was changed and the changes was updated in DataSet linked to the grid (when FilterGrid.Active is True). Call FilterUpdated (usually into OnFilterChanged event handler) to notify the filter settings in DataSet linked to the grid was changed. FilterUpdated calls OnFilterUpdated event handler to take any specific action when filter was changed in DataSet.

See also: TXCustomDBGrid.OnFilterChanged, TXCustomDBGrid.OnFilterUpdated

TXCustomDBGrid.FindColumn

Searches for a specified column object in the TXDBGridColumns object.

```
function FindColumn(const FieldName: string; FromIndex: Integer = 0):
TXColumn;
function FindColumn(const FieldName: WideString; FromIndex: Integer = 0):
TXColumn; // C++Builder/Delphi 2006, 2007 Win32 only
```

Description
Call FindColumn to determine if a specified field object is referenced in any column contained in the TXDBGridColumns object. FieldName is the name of the field stored in column's FieldName property for which to search. If FindColumn finds a field with a matching name in any column, it returns the TXColumn object for the specified field. Otherwise it returns nil. Specify FromIndex to continue searching duplicated FieldName starting from specified column index.

See also: TXCustomDBGrid.ColumnByName

TXCustomDBGrid.ForcedSequence

Indicates whether sequence numbers for database records are forced in grid.

```
function ForcedSequence: Boolean;
```

Description

This function has similar meaning to IsSequenced. ForcedSequence is True, if dgForceSequence option is included in Optionsand dataset linked to the grid is Active. Otherwise, ForcedSequence is False.

When ForcedSequence is True the grid can realize proportional scrolling and AutoNumber for any dataset. See also: UpdateSequence, RecNumber, RecCount.

See also: TXCustomDBGrid.UpdateSequence, TXCustomDBGrid.RecNumber, TXCustomDBGrid.RecCount, TXCustomDBGrid.Options

TXCustomDBGrid.ForEachDataCell

Processes all/selected data cells in the grid.

```
type
  TForEachCellScope = (fsAutoSelect, fsWholeDataSet, fsSelectedOnly);
  TForEachCellFlag = (feFirstCol, feLastCol, feFirstRow, feLastRow);
  TForEachCellFlags = set of TForEachCellFlag;
  TForEachCellProc = procedure(Column: TXColumn; Flags: TForEachCellFlags) of
    object; {* ver. 4.3 *} // For Delphi 2007 and earlier
  TForEachCellProc = reference to procedure(Column: TXColumn; Flags:
    TForEachCellFlags); {* ver. 6.2 *} // For Delphi 2009 or higher
procedure ForEachDataCell(DoProc: TForEachCellProc; Scope: TForEachCellScope =
fsAutoSelect); {* ver. 4.3 *} // For Delphi 2007 and earlier
procedure ForEachDataCell(DoProc: TForEachCellProc; Scope: TForEachCellScope =
fsAutoSelect); overload; {* ver. 6.2 *} // For Delphi 2009 or higher
```

Description

Use ForEachDataCell to process all/selected data cells in the grid. The grid calls DoProc method for each data cell determined by Scope parameter. When Scope is fsSelectedOnly the ForEachDataCell can handle all kinds of multiselection depend on MultiSelect property.

To use ForEachDataCell method you must define DoProc method to process values for each data cell. When DoProc method is called the current record in dataset linked to the grid indicates the row of processed cell and Column parameter indicates the column of processed cell. The Flag parameter allows to do additional actions when first/last row/column is processed.

The Scope parameter determines the scope of processed cells from the following values:

Value	Meaning
fsAutoSelect	When MultiSelected is True, the scope is fsSelectedOnly else fsWholeDataSet.
fsWholeDataSet	All columns in whole dataset are processed.
fsSelectedOnly	Selected rows/columns/cells are processed only.

Example

```
procedure TForm1.DoSaveToFile(Column: TXColumn; Flags: TForEachCellFlags);
const
  List: TStrings = nil;
  Text: string = '';
  ARow: Integer = 0;
begin
```



```
  if feFirstCol in Flags then
  begin
    if feFirstRow in Flags then
    begin
      FreeAndNil(List);
      List := TStringList.Create;
      ARow := 0;
    end;
    Text := '';
    Inc(ARow);
  end;
  if Column.Showing then
  begin
    if Column.AutoNumber then Text := Text + AutoNumberStr(ARow)
    else Text := Text + StringReplace(Column.EditText, ';', ' ',
[rfReplaceAll]);
    if not (feLastCol in Flags) then Text := Text + ';';
  end;
  if feLastCol in Flags then
  begin
    List.Add(Text);
    if feLastRow in Flags then
    begin
      List.SaveToFile('SaveToFile.txt');
      FreeAndNil(List);
      Text := '';
    end;
  end;
end;

procedure TForm1.SaveToFile(Sender: TObject);
begin
  XDBGrid1.ForEachDataCell(DoSaveToFile);
end;
```

See also: TXCustomDBGrid.CopyToClipboard

TXCustomDBGrid.ForEachDataCell

Processes all/selected/current data cells in the grid.

```
procedure ForEachDataCell(RowEnum: TDataRowEnum; ColEnum: TDataColEnum;
DoProc: TDataColProc); overload; {* ver. 6.2 *} // For Delphi 2009 or higher
```

Description

Use ForEachDataCell to process all/selected/current data cells in the grid. The grid calls DoProc method for each data row determined by RowEnum parameter and for each column determined by ColEnum parameter. To use ForEachDataCell method you must define DoProc method to process values for each data cell. When DoProc method is called the current record in dataset linked to the grid indicates the row of processed cell and Column parameter indicates the column of processed cell.

The ForEachDataCell method works similar to `ForEachDataRow(RowEnum, procedure(ForEachDataCol(ColEnum, DoProc)))` but it's more efficient.

See also: TXCustomDBGrid.ForEachDataRow, TXCustomDBGrid.ForEachDataCol

TXCustomDBGrid.ForEachDataCol

Processes all/selected/current columns in the grid.

```
type
  TDataColEnum = (ceAllCols, ceSelectedCols, ceSelectedColsOrAllCols,
    ceSelectedColsOrCurrent, ceCurrentCol);
  TDataColProc = reference to procedure(Column: TXColumn);

procedure ForEachDataCol(ColEnum: TDataColEnum; DoProc: TDataColProc); {* ver.
6.2 *} // For Delphi 2009 or higher
```

Description
Use ForEachDataCol to process all/selected/current columns in the grid. The grid calls DoProc method for each column determined by ColEnum parameter. To use ForEachDataCol method you must define DoProc method to process values for each column. When DoProc method is called the Column parameter indicates the processed column.

These are the possible values of ColEnum parameter:

Value	Meaning
ceAllCols	All columns in the grid.
ceSelectedCols	All selected columns in the grid.
ceSelectedColsOrAllCols	All selected columns in the grid or all columns when no column is selected.
ceSelectedColsOrCurrent	All selected columns in the grid or current column when no column is selected.
ceCurrentCol	Current column in the grid.

Example

```
function TForm1.FieldNameList(ColEnum: TDataColEnum): string;
var
  List: string;
begin
  XDBGrid1.ForEachDataCol(ColEnum,
    procedure(Column: TXColumn)
    begin
      if Column.Showing then
        List := List + ';' + Column.FieldName;
    end);
  Result := Copy(List, 2, Length(List));
end;
```

See also: TXCustomDBGrid.ForEachDataRow, TXCustomDBGrid.GetDataColEnum

TXCustomDBGrid.ForEachDataRow

Processes all/selected/current data rows in the grid.

```
type
  TDataRowEnum = (reAllRows, reSelectedRows, reSelectedRowsOrAllRows,
    reSelectedRowsOrCurrent, reCurrentRow);
  TDataRowProc = reference to procedure;

procedure ForEachDataRow(RowEnum: TDataRowEnum; DoProc: TDataRowProc); {* ver.
6.2 *} // For Delphi 2009 or higher
```

Description
Use ForEachDataRow to process all/selected/current data rows in the grid. The grid calls DoProc method

for each data row determined by RowEnum parameter. To use ForEachDataRow method you must define DoProc method to process values for each data row. When DoProc method is called the current record in dataset linked to the grid indicates the processed row.

These are the possible values of RowEnum parameter:

Value	Meaning
reAllRows	All data rows in the grid.
reSelectedRows	All selected rows in the grid.
reSelectedRowsOrAllRows	All selected rows in the grid or all data rows when no row is selected.
reSelectedRowsOrCurrent	All selected rows in the grid or current data row when no row is selected.
reCurrentRow	Current data row in the grid or nothing when data set is empty.

Example

```
function TForm1.FieldTextList(Field: TField; RowEnum: TDataRowEnum): string;
var
  List: string;
begin
  XDBGrid1.ForEachDataRow(RowEnum,
    procedure
    begin
      List := List + ';' + Field.Text;
    end);
  Result := Copy(List, 2, Length(List));
end;
```

See also: TXCustomDBGrid.ForEachDataCol, TXCustomDBGrid.GetDataRowEnum

TXCustomDBGrid.GetDataColEnum

Retrieves a column enumerator.

```
function GetDataColEnum(ColEnum: TDataColEnum): TXColumnEnumerator; {* ver. 6.2 *} // For Delphi 2009 or higher
```

Description
GetDataColEnum method retrieves a column enumerator. The ColEnum parameter determines a kind of column enumerator.

See also: TXCustomDBGrid.GetDataRowEnum, TXCustomDBGrid.ForEachDataCol

TXCustomDBGrid.GetDataRowEnum

Retrieves a bookmark enumerator.

```
function GetDataRowEnum(RowEnum: TDataRowEnum): TXBookmarkEnumerator; {* ver. 6.2 *} // For Delphi 2009 or higher
```

Description
GetDataRowEnum method retrieves a bookmark enumerator. The RowEnum parameter determines a kind of bookmark enumerator.

See also: TXCustomDBGrid.GetDataColEnum, TXCustomDBGrid.ForEachDataRow

TXCustomDBGrid.GetTotalText

Retrieves Text for total cell.

```
function GetTotalText(Column: TXColumn; Index: Integer; TextKind: TTextKind):
string; virtual;
```

Description
GetTotalText method retrieves Text to appear within total cell depend on current value of TotalResult property and other TXColumnTotal properties. The Column parameter determines column of total cell, the Index parameter corresponds to the number of total row. The TextKind parameter is reserved for future use (use tkDisplay).

See also: TXCustomDBGrid.ColGetText, TXCustomDBGrid.OnTotalGetText

TXCustomDBGrid.GotoPosition

Moves the record to which a specified bookmark points to the specified position in the grid.

```
function GotoPosition(Position: Integer; Bookmark: TBookmarkStr =
EmptyBookmark): Boolean;
function GotoPosition(Position: Integer; Bookmark: TBookmark = EmptyBookmark):
Boolean; // Delphi 2009, 2010, XE
```

Description
GotoPosition calls internally GotoBookmark method for Bookmark parameter and places the specified record in the grid as a row specified by Position parameter. When Bookmark parameter is omited the current record is placed on the Position in the grid. Value for Position parameter should be in range 0 to DataRowCount-1. The record is placed exactly on the specified position in the grid only when the record's position in DataSet allows to do this. Otherwise, the record is placed on the nearest possible Position. See also Position property.

The function return True when operation was finished successfully. When Bookmark is not valid the function return False.

See also: TXCustomDBGrid.Position, TXCustomDBGrid.DisablePosition, TXCustomDBGrid.EnablePosition, TXCustomDBGrid.PositionDisabled, TXCustomDBGrid.DataRowCount

TXCustomDBGrid.GridDisabled

Indicates whether XDBGrid component do not response on incoming events.

```
function GridDisabled: Boolean;
```

Description
Call GridDisabled to determine if the response on incoming events is currently disabled. If GridDisabled is True, XDBGrid is currently disabled by prior using DisableGrid.

See also: TXCustomDBGrid.DisableGrid, TXCustomDBGrid.EnableGrid

TXCustomDBGrid.InvertAll

Invert selection of all records or columns in the grid.

```
procedure InvertAll;
```

Description
Use InvertAll to invert selection of all records or columns in the grid. This method works only when MultiSelect is msRows or msCols.

If Options include dgExtendedSelect, the user can invert selection of all records or columns by using Alt+Ctrl+Click.

See also: TXCustomDBGrid.InvertRows, TXCustomDBGrid.SelectAll, TXCustomDBGrid.UnselectAll

TXCustomDBGrid.InvertRows

Invert selection of multiple records in the grid.

```
procedure InvertRows;
```

Description
Use InvertRows to invert selection of records from SelectionAnchor to the current record. This method works only when MultiSelect is msRows.

See also: TXCustomDBGrid.InvertAll, TXCustomDBGrid.SelectRows, TXCustomDBGrid.UnselectRows

TXCustomDBGrid.IsCustomStyle

Indicates whether grid uses custom style to draw fixed and/or data cells.

```
function IsCustomStyle(Elements: TStyleElements = []): Boolean; {* ver. 6.0 *}
// For C++Builder/Delphi XE2 or higher
```

Description
Call IsCustomStyle to determine whether grid uses custom style drawing for fixed and/or data cells. IsCustomStyle function return True when StylesEnabled is True. When Elements parameter is not omitted, the StyleElements property must also contain all Elements passed in parameter.

See also: StylesEnabled, TXCustomDBGrid.IsSystemStyle

TXCustomDBGrid.IsGridThemed

Indicates whether grid uses ThemeServices (Windows XP styles) to draw fixed cells.

```
function IsGridThemed(ForceThemed: Boolean = False){* ver. 5.0 *}: Boolean;
```

Description
Call IsGridThemed to determine whether XDBGrid uses Windows XP style drawing for fixed cells. IsGridThemed return True when ThemesEnabled return True and FixedTheme property is not ftNone or either the ForceThemed parameter is True.

See also: TXCustomDBGrid.ThemesEnabled, TXCustomDBGrid.FixedTheme, TXCustomDBGrid.GridStyle

TXCustomDBGrid.IsSystemStyle

Indicates whether grid uses system style to draw fixed and/or data cells.

```
function IsSystemStyle(Elements: TSystemStyleElements = []): Boolean; {* ver. 5.3 *} // For C++Builder/Delphi XE2 or higher
```

Description
Call IsSystemStyle to determine whether grid uses system style drawing for fixed and/or data cells. IsSystemStyle function return True when class property SystemStyleEnabled is True and StylesEnabled is False, IsVista return True, ThemesEnabled return True and FixedTheme property is not ftNone. When Elements parameter is not omitted, the SystemStyleElements property must also contain all Elements passed in parameter.

See also: StylesEnabled, ThemesEnabled, IsVista, TXCustomDBGrid.IsCustomStyle

TXCustomDBGrid.IsSystemTheme

Indicates whether system theme is used instead default fixed theme.

```
function IsSystemTheme: Boolean; {* ver. 5.32 *}
```

Description
Call IsSystemTheme to determine whether grid uses system theme drawing instead default theme for fixed cells. IsSystemTheme function return True when IsGridThemed return True and FixedTheme property is ftDefault and Windows 8 platform was detected or either Windows Vista or Windows 7 was detected when dgForceSystemTheme option is included in OptionsExt.

See also: FixedTheme, IsGridThemed, IsVista

TXCustomDBGrid.IsThemeTransparent

Indicates whether current fixed theme is drawing transparently.

```
function IsThemeTransparent: Boolean; {* ver. 5.32 *}
```

Description
Call IsThemeTransparent to determine whether current fixed theme is drawing transparently. IsThemeTransparent function return True when FixedTheme property is ftDefault and Windows 8 platform was not detected and Windows Vista or Windows 7 was detected but dgForceSystemTheme option is not included in OptionsExt. Only when IsThemeTransparent return True the FixedColor property works effectively when fixed theme is applied.

See also: FixedTheme, IsVista

TXCustomDBGrid.KeepPosition

Indicates whether the grid can keep position of current record.

```
function KeepPosition: Boolean; {* ver. 5.32 *}
```

Description
Call KeepPosition to determine whether the grid can keep position of current record. This function return True when KeyFields property is defined. KeepPosition is evaluated by RefreshDataSet and ChangeDataSetOrder methods.

See also: KeepSelection

TXCustomDBGrid.KeepSelection

Indicates whether the grid can keep current selection.

```
function KeepSelection: Boolean; {* ver. 5.32 *}
```

Description
Call KeepSelection to determine whether the grid can keep current selection. This function return True when KeyFields property is defined and dgAutoKeepSelection option is included in OptionsEx. KeepSelection is evaluated by RefreshDataSet and ChangeDataSetOrder methods.

See also: KeepPosition

TXCustomDBGrid.MouseWheelScrollRows

Indicates the number of rows that are scrolled for each notch that the mouse wheel is rotated.

```
function MouseWheelScrollRows: Integer;
```

Description
Call MouseWheelScrollRows to get number of rows that are currently scrolled for each notch that the mouse wheel is rotated. The value of this function is determined by WheelScrollRows property.

See also: TXCustomDBGrid.WheelScrollRows

TXCustomDBGrid.OrderUpdated

Triggers OnOrderUpdated event.

```
procedure OrderUpdated; virtual;
```

Description
The OrderUpdated method is called when sorting order in dataset linked to the grid was successfully changed by one of the methods designed to automatically change sorting order (ChangeDataSetOrder, ChangeIndexFields, ChangeOrderFields or ChangeSQLOrderFields) or by the method registered with using RegisterCustomChangeOrder procedure. When you write custom method to change sorting order remember to call OrderUpdated when you method successfully finished.

The OrderUpdated method triggers OnOrderUpdated event.

See also: TXCustomDBGrid.UpdateOrder, TXCustomDBGrid.OnOrderUpdated

TXCustomDBGrid.PerformDelayed

Performs delayed calculations.

```
procedure PerformDelayed: Boolean; virtual;
```

Description
Call PerformDelayed to perform all delayed calculations when dgDelayUpdateSequence, dgDelayUpdateTotals or dgDelaySelectedRows option is included in OptionsEx property. PerformDelayed always calls CheckBrowseMode first and then finalizes delayed UpdateSequnece, UpdateTotals and/or SelectedRowsChanged methods because these methods can be finalized only when DataSet linked to the grid is in dsBrowse State. You may need to call PerformDelayed (to update sequence number or total values) before Append, Insert or Edit is called outside the grid.

Notice. If you are using (X)DBNavigator with nbInsert button you should write BeforeAction (or OnBeforeInsert) event handler to call PerformDelayed before Insert.

See also: DelayInterval, TXCustomDBGrid.OptionsEx

TXCustomDBGrid.PositionDisabled

Indicates whether XDBGrid component has disabled position.

```
function PositionDisabled: Boolean;
```

Description
Call PositionDisabled to determine if XDBGrid has disabled position. If PositionDisabled is True, XDBGrid is currently disabled by prior using DisablePosition.

See also: TXCustomDBGrid.DisablePosition, TXCustomDBGrid.EnablePosition

TXCustomDBGrid.RefreshDataSet

Refresh dataset linked to the grid.

```
procedure RefreshDataSet(AKeyFields: string = ''){* ver. 5.0 *};
```

Description
Use RefreshDataSet to refresh data in the grid when data in DataSet linked to the grid changed. This method performs Close and Open on DataSet linked to the grid with using DisablePosition and EnablePosition. When AKeyFields parameter is defined, the KeyValues for current record are saved before Close and current record is restored after Open with using Locate method. When AKeyFields parameter is empty, the KeyFields property will be used instead.

Using RefreshDataSet allows to refresh data in the grid without changing the current position in the grid when KeepPosition is True and without clearing the current selection when KeepSelection is True. It's especially useful after calling ApplyUpdates method.

See also: TXCustomDBGrid.KeyFields

TXCustomDBGrid.RestoreSelection

Restore saved selection.

```
procedure RestoreSelection(AKeyFields: string = '') ; {* ver. 5.32 *}
```

Description
Call RestoreSelection to restore selection saved by SaveSelection method. AKeyFields parameter specifies the field or fields to uniquely identify records in dataset. To use more than one field, separate each field name with a semicolon. When AKeyFields parameter is empty, the KeyFields property will be used. When KeyFields property is empty, an error will be raised. The specified key fields must be of the same types as the corresponding key fields used in SaveSelection method, or the RestoreSelection method fails. This method always restores current state of SelectedRows, SelectedCols and MultiSelect properties.

This method is called automatically by RefreshDataSet and ChangeDataSetOrder procedures when KeepSelection function return True.

See also: TXCustomDBGrid.SaveSelection

TXCustomDBGrid.SaveSelection

Save current state of selection.

```
procedure SaveSelection(AKeyFields: string = '') ; {* ver. 5.32 *}
```

Description
Call SaveSelection to save current state of selection. AKeyFields parameter specifies the field or fields to uniquely identify records in dataset. To use more than one field, separate each field name with a semicolon. When AKeyFields parameter is empty, the KeyFields property will be used. When KeyFields property is empty, an error will be raised. Each call of SaveSelection procedure overrides the saved selection. This method always saves current state of SelectedRows, SelectedCols and MultiSelect properties.

This method is called automatically by RefreshDataSet and ChangeDataSetOrder procedures when KeepSelection function return True.

See also: TXCustomDBGrid.RestoreSelection

TXCustomDBGrid.ScaleColumnsBy

Scale the width and height of grid columns.

```
procedure ScaleColumnsBy(M, D: Integer; ScaleFont: Boolean = True{* ver. 6.3 *}); {* ver. 6.2 *}
```

Description

Use ScaleColumnsBy to change width and height of grid columns proportionally.

The M and D parameters define a fraction by which to scale the width of columns. The M parameter is the multiplier and the D parameter is the divisor. For example, to make a width 75% of its original size, specify the value of M as 75, and the value of D as 100 (75/100). Alternately, the same results are achieved by specifying the value of M as 3, and the value of D as 4 (3/4). Both fractions are equal and result in the width of columns being scaled by the same amount, 75%.

To scale the width of columns to be 33% larger than its previous size, specify the value of M as 133, and the value of D as 100 (133/100). Alternately, specify the value of M as 4, and the value of D as 3 (4/3), as the fraction 133/100 is approximately equal to 4/3.

Since version 6.3 all fonts used in TXDBGrid are also scaled when ScaleFont parameter is True. It allows to change columns width and rows height proportionally. {* ver. 6.3 *}

See also: TXCustomDBGrid.OptionsExt

TXCustomDBGrid.SearchCell

Searches for any text in the grid.

```
type
  TXSearchMode = (smFirst, smNext, smPrev, smLast);
function SearchCell(SearchMode: TXSearchMode; const SearchText: string;
SearchOptions: TXSearchOptions): Boolean; {* ver. 6.2 *}
```

Description

Use SearchCell to search any text in the grid. The SearchText parameter contains any text to search. The SearchOptions parameter constains options to match SearchText with data in the grid. Only soCaseSensitive and soWholeWords options are taken into account. The SearchMode determines the result of the searching. When SearchText has been found SearchCell returns True and current row and current column are set up to the cell that contains the searched text. When SearchText has not been found SearchCell returns False and current row and current column are not changed. These are the possible values of SearchMode:

Value	Meaning
smFirst	The function returns first occurence of search text in the grid.
smNext	The function returns next occurence of search text in the grid.
smPrev	The function returns previous occurence of search text in the grid.
smLast	The function returns last occurence of search text in the grid.

See also: TXCustomDBGrid.SearchGrid

TXCustomDBGrid.SearchGrid

Searches for text in the grid.

```
function SearchGrid(SearchMode: TXSearchMode): Boolean; {* ver. 6.2 *}
```

Description
Use SearchGrid to search Search.Text value in the grid with using Search.Options. The SearchGrid calls internally SearchCell method and respects soWrapAround and soSkipMessage search options. The SearchMode determines the result of the searching. When search text has been found SearchGrid returns True and current row and current column are set up to the cell that contains the searched text. When search text has not been found SearchGrid returns False and current row and current column are not changed.

See also: TXCustomDBGrid.SearchCell

TXCustomDBGrid.SelectAll

Select all records/columns/cells in the grid.

```
procedure SelectAll;
```

Description
Use SelectAll to select all records/columns/cells in the grid. This method does'nt change current value of MultiSelect property.

If Options include dgExtendedSelect, the user can select all records/columns/cells by using Alt+Shift+Click or Ctrl+A key.

See also: TXCustomDBGrid.SelectRows, TXCustomDBGrid.InvertAll, TXCustomDBGrid.UnselectAll

TXCustomDBGrid.SelectRows

Select multiple records in the grid.

```
procedure SelectRows;
```

Description
Use SelectRows to select records from SelectionAnchor to the current record. This method works only when MultiSelect is msRows.

If Options include dgExtendedSelect, the user can select multiple records by using Shift+Click.

See also: TXCustomDBGrid.SelectAll, TXCustomDBGrid.InvertRows, TXCustomDBGrid.UnselectRows

TXCustomDBGrid.SetupCurrentOrder

Sets current sorting order retrieved from dataset linked to the grid.

```
procedure SetupCurrentOrder;
```

Description
SetupCurrentOrder calls CurrentDataSetOrder function to retrieve current sorting order from DataSet linked to the grid and pass the result to SetupOrderFields method to synchronize sorting Markers with sorting order in DataSet. When dgLoadCurrentOrder is included in OptionsEx this method is called automatically when DataSet linked to grid is opened.

See also: ChangeDataSetOrder.

See also: TXCustomDBGrid.UpdateOrder, TXCustomDBGrid.SetupOrderFields, TXCustomDBGrid.CurrentDataSetOrder

TXCustomDBGrid.SetupOrderFields

Sets sorting order (OrderFields) in the grid.

```
procedure SetupOrderFields(Value: string);
procedure SetupOrderFields(Value: WideString); // C++Builder/Delphi 2006, 2007 Win32 only
```

Description
SetupOrderFields sets one or more fields separated by comma in OrderFields property. SetupOrderFields does not trigger OnOrderChanged event and can be used to synchronize between sorting order in DataSet and sorting Markers in the grid. To extract current sorting order from DataSet call CurrentDataSetOrder function.

See also: TXCustomDBGrid.OrderFields, TXCustomDBGrid.SetupCurrentOrder

TXCustomDBGrid.ShowSearchPanel

Show the search panel in the grid.

```
procedure ShowSearchPanel; {* ver. 6.2 *}
```

Description
Use ShowSearchPanel to show search panel in the grid. The user can perform this action by Ctrl+F shortcut.

See also: TXCustomDBGrid.CloseSearchPanel, TXCustomDBGrid.UpdateSearchPanel, TXCustomDBGrid.SearchPanelHeight, TXCustomDBGrid.SearchPanelVisible

TXCustomDBGrid.StretchGrid

Fits width of all columns to grid client area.

```
procedure StretchGrid;
```

Description
Use StretchGrid when StretchMode is True and you made any special changes you need call this method.

See also: TXCustomDBGrid.StretchMode, TXCustomDBGrid.StretchWidthMax, TXCustomDBGrid.StretchWidthMin

TXCustomDBGrid.SwitchDrawingOptions

Include or exclude DrawingOptions depending on State.

```
procedure SwitchDrawingOptions(Part: TGridDrawingOptions; State: Boolean); {* ver. 6.0 *}
```

Description
Use SwitchDrawingOptions to include or exclude the Part of DrawingOptions property. When State is True the Part options are included, otherwise excluded.

See also: TXCustomDBGrid.SwitchHintOptions

TXCustomDBGrid.SwitchHintOptions

Include or exclude HintOptions depending on State.

```
procedure SwitchHintOptions(Part: TXHintOptions; State: Boolean); {* ver. 6.0 *}
```

Description
Use SwitchHintOptions to include or exclude the Part of HintOptions property. When State is True the Part options are included, otherwise excluded.

See also: TXCustomDBGrid.SwitchDrawingOptions

TXCustomDBGrid.SwitchMultiSelectOptions

Include or exclude MultiSelectOptions depending on State.

```
procedure SwitchMultiSelectOptions(Part: TMultiSelectOptions; State: Boolean); {* ver. 6.0 *}
```

Description
Use SwitchMultiSelectOptions to include or exclude the Part of MultiSelectOptions property. When State is True the Part options are included, otherwise excluded.

See also: TXCustomDBGrid.SwitchResizeOptions

TXCustomDBGrid.SwitchOptions

Include or exclude Options depending on State.

```
procedure SwitchOptions(Part: TXDBGridOptions; State: Boolean); {* ver. 6.0 *}
```

Description
Use SwitchOptions to include or exclude the Part of Options property. When State is True the Part options are included, otherwise excluded.

See also: TXCustomDBGrid.SwitchOptionsEx, TXCustomDBGrid.SwitchOptionsExt

TXCustomDBGrid.SwitchOptionsEx

Include or exclude OptionsEx depending on State.

```
procedure SwitchOptionsEx(Part: TXDBGridOptionsEx; State: Boolean); {* ver. 6.0 *}
```

Description
Use SwitchOptionsEx to include or exclude the Part of OptionsEx property. When State is True the Part options are included, otherwise excluded.

See also: TXCustomDBGrid.SwitchOptions, TXCustomDBGrid.SwitchOptionsExt

TXCustomDBGrid.SwitchOptionsExt

Include or exclude OptionsExt depending on State.

```
procedure SwitchOptionsExt(Part: TXDBGridOptionsExt; State: Boolean); {* ver. 6.0 *}
```

Description
Use SwitchOptionsExt to include or exclude the Part of OptionsExt property. When State is True the Part options are included, otherwise excluded.

See also: TXCustomDBGrid.SwitchOptions, TXCustomDBGrid.SwitchOptionsEx

TXCustomDBGrid.SwitchResizeOptions

Include or exclude ResizeOptions depending on State.

```
procedure SwitchResizeOptions(Part: TXResizeOptions; State: Boolean); {* ver. 6.0 *}
```

Description
Use SwitchResizeOptions to include or exclude the Part of ResizeOptions property. When State is True the Part options are included, otherwise excluded.

See also: TXCustomDBGrid.SwitchMultiSelectOptions

TXCustomDBGrid.SwitchStyleElements

Include or exclude StyleElements depending on State.

```
procedure SwitchStyleElements(Part: TStyleElements; State: Boolean); {* ver. 6.0 *} // For C++Builder/Delphi XE2 or higher
```

Description
Use SwitchStyleElements to include or exclude the Part of StyleElements property. When State is True the Part elements are included, otherwise excluded.

See also: TXCustomDBGrid.SwitchSystemStyleElements

TXCustomDBGrid.SwitchSystemStyleElements

Include or exclude SystemStyleElements depending on State.

```
procedure SwitchSystemStyleElements(Part: TSystemStyleElements; State:
Boolean); {* ver. 6.0 *} // For C++Builder/Delphi XE2 or higher
```

Description
Use SwitchSystemStyleElements to include or exclude the Part of SystemStyleElements property. When State is True the Part elements are included, otherwise excluded.

See also: TXCustomDBGrid.SwitchStyleElements

TXCustomDBGrid.Synchronize

Fits width of all columns to width of MasterGrid's columns.

```
procedure Synchronize;
```

Description
Use Synchronize when MasterGrid is assigned and you made any special changes you need call this method.

See also: TXCustomDBGrid.MasterGrid, TXCustomDBGrid.ManageGrid, TXColumn.Synchronize

TXCustomDBGrid.ThemesEnabled

Indicates whether ThemeServices (Windows XP style) is available and enabled.

```
function ThemesEnabled: Boolean;
```

Description
Call ThemesEnabled to determine whether ThemeServices is available and enabled. ThemesEnabled return True only on Windows XP platform when XP Manifest is present, otherwise ThemesEnabled return False. See also IsGridThemed.

See also: TXCustomDBGrid.IsGridThemed, TXCustomDBGrid.FixedTheme, TXCustomDBGrid.GridStyle

TXCustomDBGrid.UnselectAll

Unselect all records/columns/cells in the grid.

```
procedure UnselectAll;
```

Description
Use UnselectAll to unselect all records/columns/cells in the grid. This method calls internally ClearSelection. See also: MultiSelect.

If Options include dgExtendedSelect, the user can unselect all records/columns/cells by using Alt+Click.

See also: TXCustomDBGrid.UnselectRows, TXCustomDBGrid.InvertAll, TXCustomDBGrid.SelectAll

TXCustomDBGrid.UnselectRows

Unselect multiple records in the grid.

```
procedure UnselectRows;
```

Description
Use UnselectRows to unselect records from SelectionAnchor to the current record. This method works only when MultiSelect is msRows.

See also: TXCustomDBGrid.UnselectAll, TXCustomDBGrid.InvertRows, TXCustomDBGrid.SelectRows

TXCustomDBGrid.UpdateFilter

Updates filter settings in dataset linked to the grid.

```
procedure UpdateFilter; {* ver. 6.0 *}
```

Description
UpdateFilter changes filter settings in DataSet linked to the grid according to the current state of filter settings in the grid. UpdateFilter calls internally ChangeDataSetFilter to change properties in DataSet. You not need to call this method directly.

See also: TXCustomDBGrid.ChangeDataSetFilter, TXCustomDBGrid.FilterUpdated, TXCustomDBGrid.OnFilterUpdated

TXCustomDBGrid.UpdateListItems

Updates the auto-loaded list items for the grid's columns.

```
procedure UpdateListItems(CheckAutoUpdate: Boolean = False); {* ver. 4.3 *}
```

Description
UpdateListItems updates PickList property for each column when poAutoLoadList option is included in PickOptions. When CheckAutoUpdate parameter is True the method is performed only when dgAutoUpdateListItems is included in OptionsExt. The grid calls automatically this method (with CheckAutoUpdate = True) each time when data in the grid have been changed. When dgAutoUpdateListItems is included in OptionsExt the auto-loaded list items are always updated, but this method may be called to often. You can minimize the number of calls by including dgDelayUpdateListItems option in OptionsExt.

When dgAutoUpdateListItems is not included in OptionsEx, this method is called only when DataSet is opened. Then, you must yourself call UpdateListItems when you need to update auto-loaded list items especially after EnableControls and EnablePosition. You should also call yourself UpdateListItems after DataSet's method: Post, Delete, Refresh or when filter or range in dataset has been changed and when this changes may have an influence on list items.

Each time when UpdateListItems is finished the OnListItemsUpdated event is triggered.

See also: TXCustomDBGrid.InUpdateListItems, TXCustomDBGrid.OnListItemsUpdated

TXCustomDBGrid.UpdateOrder

Modify sorting order in datset linked to the grid or update sorting markers in the grid.

```
procedure UpdateOrder(CheckAutoUpdate: Boolean = False);
```

Description
UpdateOrder synchronizes sorting order in dataset linked to the grid with grid's OrderFields and sorting Markers. This method is effective only when linked dataset is registered to automatically change order. All standard DataSet descendats (except unidirectional) are registered by default to automatically change order (ADO, BDE, CDS, DBX, IBX). To register other typical or specific DataSets use RegisterChangeOrder or RegisterCustomChangeOrder procedures.

When CheckAutoUpdate parameter is True the method is performed only when dgLoadCurrentOrder or dgAutoUpdateOrder is included in OptionsEx.
When dgLoadCurrentOrder is included in OptionsEx the OrderFields (and sorting markers) are initialized on the basis of current DataSet's order by calling SetupCurrentOrder. When dgLoadCurrentOrder is not included and dgAutoUpdateOrder is included in OptionsEx sorting order in DataSet linked to the grid is automatically modified on the basis of OrderFields property by calling ChangeDataSetOrder.

When CheckAutoUpdate parameter is False the method always performs ChangeDataSetOrder to modify order in DataSet.

See also: TXCustomDBGrid.OnOrderChanged, TXCustomDBGrid.OnOrderUpdated, TXCustomDBGrid.SetupCurrentOrder, TXCustomDBGrid.ChangeDataSetOrder

TXCustomDBGrid.UpdateSearchPanel

Update bounds of the search panel in the grid.

```
procedure UpdateSearchPanel; {* ver. 6.2 *}
```

Description
Use UpdateSearchPanel to update bounds of the search panel in the grid. You not need to call this method directly.

See also: TXCustomDBGrid.CloseSearchPanel, TXCustomDBGrid.ShowSearchPanel, TXCustomDBGrid.SearchPanelHeight, TXCustomDBGrid.SearchPanelVisible

TXCustomDBGrid.UpdateSequence

Updates the sequence numbers for database records linked to grid.

```
procedure UpdateSequence(CheckAutoUpdate: Boolean = False);
```

Description
UpdateSequence updates RecNumber and RecCount property when ForcedSequence is True. When CheckAutoUpdate parameter is True the method is performed only when dgAutoUpdateSequence is included in OptionsEx. The grid calls automatically this method (with CheckAutoUpdate = True) each time when data in the grid changed. When dgAutoUpdateSequence is included in OptionsEx the sequence number is always updated, but this method may be called to often. You can minimize the number of calls by including dgDelayUpdateSequence option in OptionsEx.

When dgAutoUpdateSequence is not included in OptionsEx, this method is called automatically (with CheckAutoUpdate = False) only when dataset linked to the grid is opened. Then, you must yourself call UpdateSequence especially after EnableControls or EnablePosition (when ControlsDisabled or PositionDisabled is True the grid is'nt notified about changes in DataSet). You should also call yourself UpdateSequence after DataSet's method: Locate, FindKey, FindNearest, GotoKey, GotoNearest, GotoBookmark, Insert, Delete, Refresh or when index, filter or range in dataset has been changed and when this changes may have an influence on sequence number.

All navigation actions made by user directly in the grid update sequence internally in the grid. All sequence updates are always stored only in the grid. In the linked dataset nothing is changed. Each time when UpdateSequence is finished the OnSequenceUpdated event is triggered.

See also: TXCustomDBGrid.ForcedSequence, TXCustomDBGrid.RecNumber, TXCustomDBGrid.RecCount, TXCustomDBGrid.Options, TXCustomDBGrid.InUpdateSequence, TXCustomDBGrid.OnSequenceUpdated, TXCustomDBGrid.PerformDelayed

TXCustomDBGrid.UpdateTotals

Updates the calculated values in totals cells.

```
procedure UpdateTotals(CheckAutoUpdate: Boolean = False);
```

Description
UpdateTotals updates Value property for each total cell when TotalResult is trCalcValue. When CheckAutoUpdate parameter is True the method is performed only when dgAutoUpdateTotals is included in OptionsEx. The grid calls automatically this method (with CheckAutoUpdate = True) each time when data in the grid or selected rows have been changed. When dgAutoUpdateTotals is included in OptionsEx the calculated values are always updated, but this method may be called to often. You can minimize the number of calls by including dgDelayUpdateTotals option in OptionsEx.

When dgAutoUpdateTotals is not included in OptionsEx, this method is never called automatically. Then, you must yourself call UpdateTotals when you need to recalculate totals values especially after EnableControls, EnablePosition or when SelectedRowsChanged. You should also call yourself UpdateTotals after DataSet's method: Post, Delete, Refresh or when filter or range in dataset has been changed and when this changes may have an influence on totals values.

Each time when UpdateTotals is finished the OnTotalsUpdated event is triggered.

See also: TXCustomDBGrid.InUpdateTotals, TXCustomDBGrid.OnTotalCalcQuery, TXCustomDBGrid.OnTotalCalcValue, TXCustomDBGrid.OnTotalsUpdated, TXCustomDBGrid.PerformDelayed

TXCustomDBGrid.UpdateTreeView

Update tree view in the grid.

```
procedure UpdateTreeView(CheckAutoUpdate: Boolean = False); {* ver. 7.0 *}
```

Description
Use UpdateTreeView to update tree view in the grid. You need to call this method only when TreeView.Active is True and TreeView.AutoUpdate is False and key data in the grid changed.

See also: TXCustomDBGrid.TreeView, TXDBGridTreeView.Active, TXDBGridTreeView.AutoUpdate

TXCustomDBGrid.OnCalcBoldDays

Occurs whenever a new month is displayed in the calendar (InplaceEditor).

```
type
  TCalcBoldDaysEvent = procedure(Sender: TObject; Date: DateTime; var BoldDay:
    Boolean) of object;
property OnCalcBoldDays: TCalcBoldDaysEvent;
```

Description
Use the OnCalcBoldDays event to initialize the display properties of a month. In particular, OnCalcBoldDays allows applications to bold individual days in the calendar (such as holidays).

Default value of BoldDay is False (for every Date). To specify that specific dates are to be bolded, set BoldDay to True. To specify that all sundays are to be bolded you can use: BoldDay := DayOfWeek(Date) = 1;

Notice. Be careful, OnCalcBoldDays event is fired for every showing day in calendar.

See also: TXColumn.ButtonStyle, TXColumn.ListOptions

TXCustomDBGrid.OnCalcImageIndex

Occurs when the grid needs an index to draw bitmap from Images.

```
type
  TCalcImageIndexEvent = procedure(Sender: TObject; Column: TXColumn; var
    Index: Integer) of object;
property OnCalcImageIndex: TCalcImageIndexEvent;
```

Description
Write an OnCalcImageIndex event handler to change default index for the Images property. OnCalcImageIndex event is fired for all columns that have assigned Images property. Default value for Index is depend on field's value. When Column parameter is Nil you may calculate custom Index for IndicatorImages based upon current record state.

This event can be also fired for column's dropdown list (Sender is TXDBPopupDataList object) and then the Column parameter points to the list's column with Images. When Sender is TXDBPopupDataList then TXDBPopupDataList(Sender).ParentColumn points to the column in the grid. {* ver. 6.5 *}

This event can be also fired for column's filter list (Sender is TXDBPopupFilterList object) and then the Column parameter points to the filter list's column with Images. When Sender is TXDBPopupFilterList then TXDBPopupFilterList (Sender).ParentColumn points to the column in the grid. {* ver. 6.5 *}

Images are visible on dropdown list and filter list only when loShowImages option is included in ListOptions property. {* ver. 6.5 *}

See also: TXCustomDBGrid.IndicatorImages, TXCustomDBGrid.Indicators, TXColumn.Images

TXCustomDBGrid.OnCellDblClick

Occurs when the user double-clicks the left mouse button in one of the data cells.

```
type
  TDBGridClickEvent = procedure (Column: TXColumn) of object;
property OnCellDblClick: TDBGridClickEvent; {* ver. 4.4 *}
```

Description
Write an OnCellDblClick event handler to take specific action when the user double-clicks in one of the

data cells of the data-aware grid (similar to OnCellClick). The Column parameter is the TXColumn object that corresponds to the column where the mouse was when the user double-clicked the left mouse button. When OnCellDblClick event occurs, OnDblClick event will not occur.

OnCellDblClick will not occur when the user double-clicks in indicator, title or total cell of the column or in the blank area. In that case the standard OnDblClick event will occur.

When OnCellDblClick event is not assigned, OnDblClick event will always occur when the user double-clicks in any area of the grid (including data cells).

See also: TCustomDBGrid.OnCellClick, TControl.OnDblClick

TXCustomDBGrid.OnCellExpand

Occurs when the user clicks any data cell for the column with ExpandBox.

```
type
  TDBGridClickEvent = procedure (Column: TXColumn) of object;
property OnCellExpand: TDBGridClickEvent; {* ver. 6.6 *}
```

Description

Write an OnCellExpand event handler to take specific action when the user clicks any data cell for the column with ExpandBox. The Column parameter is the TXColumn object that corresponds to the column where the mouse was when the user clicked the left mouse button. When OnCellExpand event occurs, OnCellClick event will not occur. Here is a simple example, how to write OnCellExpand event handler:

```
procedure TMainForm.CustomerDBGridCellExpand(Column: TXColumn);
var
  OrdersForm: TOrdersForm;
  P: TPoint;
begin
  OrdersForm := TOrdersForm.Create(Application);

  P := CustomerDBGrid.CellExpandPoint(Column, OrdersForm.Width);
  OrdersForm.Left := P.X;
  OrdersForm.Top := P.Y;

  OrdersForm.OrdersDBGrid.BiDiMode := CustomerDBGrid.BiDiMode;
  OrdersForm.OrdersDBGrid.FixedStyle := CustomerDBGrid.FixedStyle;
  OrdersForm.OrdersDBGrid.FixedTheme := CustomerDBGrid.FixedTheme;
  OrdersForm.OrdersDBGrid.Gradient := CustomerDBGrid.Gradient;

  OrdersForm.OrdersCDS.IndexFieldNames := 'CustNo';
  OrdersForm.OrdersCDS.MasterSource := CustomerDS;
  OrdersForm.OrdersCDS.MasterFields := 'CustNo';

  OrdersForm.ShowModal;
  OrdersForm.Release
end;
```

See also: TXColumn.ExpandBox, TXColumn.ExpandBoxToggle, TXColumn.ToggleExpandBox

TXCustomDBGrid.OnCellHint

Occurs when the mouse pauses over a cell that can display a Help Hint.

```
type
  THintCell = (hcTitle, hcHeader, hcFiller, hcEditor, hcData, hcIndicator,
    hcTotal);
  TCellHintEvent = procedure(Sender: TObject; Column: TXColumn; Index:
    Integer; HintCell: THintCell; var HintStr: string) of object;
property OnCellHint: TCellHintEvent;
```

Description

Use OnCellHint to override the default hint that appears when the user pauses the mouse over a cell.

In the event handler, use HintStr to override default value when displaying the hint string in a popup window. Set empty HintStr when the hint should not be shown at all. HintCell indicates kind of cell for column specified in Column parameter. When HintCell is hcHeader, an Index specifies number of row for column's header.

See also: TXCustomDBGrid.HintOptions, TXCustomDBGrid.OnCellHint, TXColumn.EditorHint, TXColumnTitle.Hint, TXColumnTitle.HeaderHint

TXCustomDBGrid.OnColExpand

Occurs when the expandable column is expanded or contracted.

```
type
  TDBGridClickEvent = procedure(Column: TXColumn) of object;
property OnColExpand: TDBGridClickEvent;
```

Description

Write an OnColExpand event to perform special processing when Expandable column is expanded or contracted. Current state of this column is determined by Expanded property.

See also: TColumn.Expanded, TColumn.Expandable, TADTField

TXCustomDBGrid.OnColGetText

Occurs when the grid retrieves Text from dataset.

```
type
  TColGetTextEvent = procedure(Column: TXColumn; var Text: string; TextKind:
    TTextKind) of object;
property OnColGetText: TColGetTextEvent;
```

Description

Write an OnColGetText event handler to change Text retrieved from dataset. OnColGetText event is called by ColGetText method.

See also: TXCustomDBGrid.ColGetText, TXCustomDBGrid.OnColSetText

TXCustomDBGrid.OnColSetText

Occurs when the grid stores Text to dataset.

```
type
  TColSetTextEvent = procedure(Column: TXColumn; var Text: string) of object;
property OnColSetText: TColSetTextEvent;
```

Description
Write an OnColSetText event handler to change Text stored to dataset. OnColSetText event is called by ColSetText method.

See also: TXCustomDBGrid.ColSetText, TXCustomDBGrid.OnColGetText

TXCustomDBGrid.OnColTextChanged

Occurs when edited text was changed and stored to dataset.

```
property OnColTextChanged: TNotifyEvent; {* ver. 6.0 *}
```

Description
Write an OnColTextChanged event handler to perform any action when edited text was changed and stored to dataset. You can retrieve the changed text by using TXDBGrid(Sender).SelectedColumn.EditText. This event occurs when Inplace Editor stores new text to dataset. The dataset is then in dsEdit/dsInsert mode.

See also: TXCustomDBGrid.OnColGetText, TXCustomDBGrid.OnColSetText

TXCustomDBGrid.OnColumnMoving

Occurs when a column is about to be moved.

```
type
  TColumnMovingEvent = procedure(Sender: TObject; FromIndex, ToIndex: Longint;
    var Accept: Boolean) of object;
property OnColumnMoving: TColumnMovingEvent;
```

Description
Write an OnColumnMoving event handler to selectively prevent selection from moving column to specific places.

The OnColumnMoving is called when the selection is about to be moved from one place to another. Set Accept to False, to prevent selection from moving to a new place.

See also: TCustomDBGrid.OnColumnMoved

TXCustomDBGrid.OnColumnResize

Occurs when one or more columns is resized.

```
property OnColumnResize: TNotifyEvent;
```

Description
Use OnColumnResize to perform special processing when any column is resized. Read ResizedColumn property to determine which column has been lately resized.

See also: TXCustomDBGrid.OnColumnScroll, TXCustomDBGrid.OnRowResize

TXCustomDBGrid.OnColumnScroll

Occurs when the user scrolls columns with the mouse or keyboard.

```
type
  TColumnScrollEvent = procedure(Sender: TObject; LeftIndex: Longint) of
    object;
property OnColumnScroll: TColumnScrollEvent;
```

Description
Use OnColumnScroll to perform special processing when columns are scrolled. The LeftIndex parameter contains index of first visible column (except fixed columns).

See also: TXCustomDBGrid.OnColumnResize, TXCustomDBGrid.OnRowResize

TXCustomDBGrid.OnExpandClick

Occurs when the user presses the expand button in a column title.

```
type
  TDBGridClickEvent = procedure(Column: TXColumn) of object;
property OnExpandClick: TDBGridClickEvent;
```

Description
Write an OnExpandClick event handler to bring up an appropriate list or dialog or drop-down a menu when the user presses the expand button in a column title. This event occurs only when ExpandStyle is cesDropDown or cesDropDownMenu and DropDownMenu is'nt assigned. See also: OnColExpand.

See also: TXCustomDBGrid.OnColExpand, TXCustomDBGrid.OnEditButtonClick

TXCustomDBGrid.OnFillerClick

Occurs when the user releases the mouse in title of indicator.

```
property OnFillerClick: TNotifyEvent;
```

Description
Write an OnFillerClick event handler to take specific action when the user clicks in title of indicator (Filler).

See also: TXCustomDBGrid.OnHeaderClick, TXCustomDBGrid.OnIndicatorClick

TXCustomDBGrid.OnFilterChanged

Occurs when FilterList or FilterText changed.

```
property OnFilterChanged: TNotifyEvent; {* ver. 6.0 *}
```

Description
Write an OnFilterChanged event handler to self handle FilterList and/or FilterText property. The event is fired only when FilterGrid.AutoUpdate is False. Otherwise, the internal filter changes are handled by the grid.

See also: TXCustomDBGrid.OnFilterUpdated

TXCustomDBGrid.OnFilterRecord

Points to OnFilterRecord event handler stored in DataSet linked to the grid.

```
property OnFilterRecord: TFilterRecordEvent; {* ver. 6.0 *}
```

Description
OnFilterRecord property points to TDataSet.OnFilterRecord event handler stored in DataSet linked to TXDBGrid. When the OnFilterRecord property is changed in TXDBGrid the new event handler is stored in DataSet property, but the change is controlled by TXDBGrid. If KeyFields are defined, the current record is restored, if the record is still available in filtered record set. If dgAutoKeepSelection option is included in OptionsEx and grid has multiselected records, still visible selected records are restored.

See also: TXCustomDBGrid.Filter, TXCustomDBGrid.Filtered

TXCustomDBGrid.OnFilterUpdated

Occurs when filter confition in DataSet changed.

```
property OnFilterUpdated: TNotifyEvent; {* ver. 6.0 *}
```

Description
Write an OnFilterUpdated event handler to take specific action when filter condition in DataSet linked to the grid was changed. OnFilterUpdated event is triggered by FilterUpdated method when DataSet linked to the grid is automatically updated by the grid (FilterGrid.AutoUpdate is True) or when you call FilterUpdated in your OnFilterChanged event handler.

See also: TXCustomDBGrid.FilterUpdated, TXCustomDBGrid.OnFilterChanged

TXCustomDBGrid.OnHeaderClick

Occurs when the user releases the mouse in one of the title headers.

```
type
  THeaderClickEvent = procedure(Column: TXColumn; Index: Integer) of object;
property OnHeaderClick: THeaderClickEvent;
```

Description
Write an OnHeaderClick event handler to take specific action when the user clicks in one of the title headers. The Column parameter is the TColumn object that corresponds to the column where the mouse was when the user released the left mouse button. The Index parameter contains number of header's row.

See also: TXCustomDBGrid.OnFillerClick, TXCustomDBGrid.OnIndicatorClick

TXCustomDBGrid.OnIndicatorClick

Occurs when the user releases the mouse in one of the cells of indicator.

```
property OnIndicatorClick: TNotifyEvent;
```

Description
Write an OnIndicatorClick event handler to take specific action when the user clicks in one of the cells of indicator.

See also: TXCustomDBGrid.OnFillerClick, TXCustomDBGrid.OnHeaderClick

TXCustomDBGrid.OnLayout

Occurs when the column layout is retrieved or modified.

```
type
  TXDBGridLayoutEvent = procedure(Sender: TObject; Layout: TStrings; Apply:
    Boolean) of object;
property OnLayout: TXDBGridLayoutEvent;
```

Description
OnLayout event occurs when the Layout property is reading or setting. Write OnLayout event handler to add additional XDBGrid's properties to Layout parameter when Apply is False and set additional XDBGrid's properties on the basis of Layout parameter when Apply is True. Layout is a TStrings object designed for store and retrieve column layout settings as name-value pairs. For more information on name-value pairs, refer to Values property.

For example:

```
with TXDBGrid(Sender) do
  if not Apply then Layout.Add('FixedStyle='+IntToStr(Ord(FixedStyle)))
  else FixedStyle := TFixedStyle(StrToIntDef(Layout.Values['FixedStyle'],
Ord(FixedStyle)));
```

See also: TXCustomDBGrid.Settings, TXDBGridSettings.Layout

TXCustomDBGrid.OnListCloseUp

Occurs when the editor's drop-down list has been closed.

```
type
  TListCloseUpEvent = procedure (Column: TXColumn; List: TObject; var Value:
    Variant; var Accept: Boolean) of object;
property OnListCloseUp: TListCloseUpEvent; {* ver. 4.3 *}
```

Description
Write an OnListCloseUp event handler to take specific action when the editor's list has been closed. The Column parameter indicates the grid's column for the list. The List parameter holds the editor's list. Look at OnListDropDown to read, how to use List.

The Value parameter contains the value selected by the user. The Accept parameter contains the user's choice result. When Accept is True, the new Value was accepted by the user, when Accept is False, the user cancelled the choice. You can change the user's choice by modifying the Value and/or the Accept parameter.

See also: TXCustomDBGrid.OnListDropDown

TXCustomDBGrid.OnListDropDown

Occurs when the editor's drop-down list is dropped-down.

```
type
  TXDataEditor = (dePickList, deDataList, deGridList, deCalendar,
    deCalculator);  {* ver. 6.0 *}
  TListDropDownEvent = procedure (Column: TXColumn; List: TObject) of object;
property OnListDropDown: TListDropDownEvent; {* ver. 4.3 *}
```

Description
Write an OnListDropDown event handler to take specific action when the editor's list is dropped-down. OnListDropDown is triggered when the editor's list is ready to appear on the screen, but it's just not visible. The Column parameter indicates the grid's column for the list. The List parameter holds the editor's list object. The type of List depends on ButtonStyle value for the column.

The class of List depends on internal value of DataEditor. Before using List object you must check is the List of appropriate class. You have not an access to the final classes of the List. You can use only the ancestor class of the List. If you want to modify the List object, you are doing it on your own risk. Remember, you must restore your changes in OnListCloseUp event handler. {* ver. 6.0 *}

Starting from version 6.0 each drop-down list is derived from TXDBGridDropDownList class and declared in XDBDataList unit: TXDBGridDataList, TXDBGridPickList, TXDBGridCalculator, TXDBGridCalendar. If you want to modify the List object, you are still doing it on your own risk. {* ver. 6.0 *}

See also: TXCustomDBGrid.OnListCloseUp

TXCustomDBGrid.OnListItemsUpdated

Occurs when the auto-loaded list items have been updated.

```
property OnListItemsUpdated: TNotifyEvent; {* ver. 4.3 *}
```

Description
Write an OnListItemsUpdated event handler to take specific action when the auto-loaded PickList items have been updated. OnListItemsUpdated is triggered when UpdateListItems is finished and all auto-loaded PickList's are updated.

See also: TXCustomDBGrid.UpdateListItems

TXCustomDBGrid.OnOrderChanged

Occurs when OrderFields (sorting markers) changed.

```
property OnOrderChanged: TNotifyEvent;
```

Description
Write an OnOrderChanged event handler to synchronize sorting markers in the grid and sorting order in DataSet linked to the grid. When dgAutoUpdateOrder is included in OptionsEx and DataSet linked to the grid is registered to automatically change order, this event is not triggered. Instead, the ChangeDataSetOrder is automatically performed.

When you need to modify OrderFields before calling ChangeDataSetOrder you should to exclude dgAutoUpdateOrder from OptionsEx and call ChangeDataSetOrder in this event handler. Then, you can pass modified value of OrderFields as parameter for ChangeDataSetOrder method.

When your DataSet linked to the grid is not registered to automatically change order you can register it by using RegisterChangeOrder or RegisterCustomChangeOrder or either you can call directly in this event one of methods designed to change order in typical DataSets: ChangeIndexFields, ChangeOrderFields or ChangeSQLOrderFields.

If non of them methods is appropriate to change order in your DataSet, you should perform changing order in DataSet in this event or you can write custom method and register it with using RegisterCustomChangeOrder procedure to automatically change order in linked DataSet.

See also: KeyFields property, OnOrderUpdated event.

See also: TXCustomDBGrid.OrderFields, TXColumnTitle.ToggleMarker, TXCustomDBGrid.UpdateOrder, TXCustomDBGrid.OnOrderUpdated

TXCustomDBGrid.OnOrderUpdated

Occurs when the sorting order in dataset was changed.

```
property OnOrderUpdated: TNotifyEvent;
```

Description
Write an OnOrderUpdated event handler to take specific action when sorting order in dataset linked to the grid was changed. OnOrderUpdated event is triggered by OrderUpdated method when dataset linked to the grid is registerd to automatically change order and dgAutoUpdateOrder option is included in OptionsEx or when you call directly in code one of the methods designed to change sorting order (ChangeDataSetOrder, ChangeIndexFields, ChangeOrderFields or ChangeSQLOrderFields) and changing order was successfully finished.

See also: OnOrderChanged event.

See also: TXCustomDBGrid.UpdateOrder, TXCustomDBGrid.OnOrderChanged

TXCustomDBGrid.OnPaintColumnCell

Occurs when the grid needs to paint a cell by using DefaultDrawing.

```
type
  TSelection = (slCellSelected, slRowSelected, slMultiSelected, slCellFixed,
    slRowStriped);
  TSelections = set of TSelection;
  TPaintColumnCellEvent = procedure(Sender: TObject; const Rect: TRect;
    DataCol: Integer; Column: TXColumn; Highlight: Boolean; Selections:
    TSelections; var Color: TColor; Font: TFont; var Image: TPersistent) of
    object;
property OnPaintColumnCell: TPaintColumnCellEvent;
```

Description
Write an OnPaintColumnCell event handler to modify default drawing for the data in the cells of the grid.

The Rect parameter indicates the location of the cell on the canvas. The DataCol parameter is the index of the column in the Columns array. The Column parameter is the TXColumn object that describes the display attributes and field binding for the cell. The Highlight parameter indicates whether the cell is highlighted. The Selections parameter indicates whether the cell is selected (slCellSelected), whether the cell is in selected row (current record) (slRowSelected), whether the cell is in one of multiselected rows (slMultiSelected) or whether the cell is fixed (slCellFixed). The slRowStriped value is included in Selections only when StripeColor is not clNone and when the cell is in even data row. Color, Font and Image parameters contain default values that may be simple modified by developer depend on current data values.

See also: TCustomDBGrid.OnDrawColumnCell

TXCustomDBGrid.OnRowExpand

Occurs when the tree view row has been expanded or collapsed.

```
type
  TRowExpandEvent = procedure(Sender: TObject; Expand: Boolean) of object;
property OnRowExpand: TRowExpandEvent; {* ver. 7.0 *}
```

Description

Write an OnRowExpand event to perform special processing when tree view row has been expanded or collapsed. When Expand parameter is True, the current row has been expanded. When Expand parameter is False, the current row has been collapsed.

See also: TXDBGridTreeView.IsRowExpanded, TXDBGridTreeView.IsRowExpandable, TXCustomDBGrid.OnRowExpanding

TXCustomDBGrid.OnRowExpanding

Occurs when the tree view row is being expanded or collapsed.

```
type
  TRowExpandingEvent = procedure(Sender: TObject; Expand: Boolean; var Allow:
    Boolean) of object;
property OnRowExpanding: TRowExpandingEvent; {* ver. 7.0 *}
```

Description

Write an OnRowExpanding event to perform special processing when tree view row is being expanded or collapsed. When Expand parameter is True, the current row is being expanded. When Expand parameter is False, the current row is being collapsed. You can set parameter Allow to False, to stop row expanding/collapsing.

See also: TXDBGridTreeView.IsRowExpanded, TXDBGridTreeView.IsRowExpandable, TXCustomDBGrid.OnRowExpand

TXCustomDBGrid.OnRowResize

Occurs when rows in the grid are resized.

```
property OnRowResize: TNotifyEvent;
```

Description

Use OnRowResize to perform special processing when rows in the grid are resized.

See also: TXCustomDBGrid.OnColumnScroll, TXCustomDBGrid.OnColumnResize

TXCustomDBGrid.OnRowSelect

Occurs when the rows selection changed.

```
type
  TSelectAction = (saUnselect, saSelect, saClear, saDelete, saRefresh);
  TRowSelectEvent = procedure(Sender: TObject; Action: TSelectAction) of
    object;
property OnRowSelect: TRowSelectEvent;
```

Description
Write an OnRowSelect event to perform special processing when the rows selection changed. Action determines what kind of changes are made. When Action is saUnselect or saSelect current record has been changed. When Action is saClear, saDelete or saRefresh all selected rows have been changed. See also appropriate methods of TXBookmarkList. Caution: this event can be called very often.

See also: TXCustomDBGrid.SelectedRows, TXBookmarkList.CurrentRowSelected

TXCustomDBGrid.OnSelectedColsChanged

Occurs when the selected columns changed.

```
property OnSelectedColsChanged: TNotifyEvent; {* ver. 4.3 *}
```

Description
Write an OnSelectedColsChanged event handler to take specific action when SelectedCols have been changed in the grid. This event is triggered after group of changes were made is SelectedCols (eg. when SelectAll was performed the event is triggered once (when ColsSelection is True) - after all changes in SelectedCols list. Use BeginSelect & EndSelect statements when group of changes is made in SelectedCols/SelectedRows list directly in your code. To minimize numbers of calls for this event the dgDelaySelectedRows option should be included in OptionsEx. This option is common for events: OnSelectedRowsChanged, OnSelectedColsChanged, OnSelectionChanged.

Note. In some cases (eg. when MultiSelect is msCells) after one group of changes the events may are triggered in order: OnSelectedRowsChanged, OnSelectedColsChanged, OnSelectionChanged. You can check RowsChanged flag to determine the OnSelectedRowsChanged event was triggered prior.

See also: TXCustomDBGrid.OnSelectionChanged, TXCustomDBGrid.OnSelectedRowsChanged, TXCustomDBGrid.ColsChanged, , TXCustomDBGrid.RowsChanged, TXCustomDBGrid.PerformDelayed

TXCustomDBGrid.OnSelectedRowsChanged

Occurs when the selected rows changed.

```
property OnSelectedRowsChanged: TNotifyEvent;
```

Description
Write an OnSelectedRowsChanged event handler to take specific action when SelectedRows have been changed in the grid. This event is triggered after group of changes were made is SelectedRows (eg. when SelectAll was performed the event is triggered once (when RowsSelection is True) - after all changes in SelectedRows list). Use BeginSelect & EndSelect statements when group of changes is made in SelectedRows/SelectedCols list directly in your code. To minimize numbers of calls for this event the dgDelaySelectedRows option should be included in OptionsEx. This option is common for events: OnSelectedRowsChanged, OnSelectedColsChanged, OnSelectionChanged.

Note. In some cases (eg. when MultiSelect is msCells) after one group of changes the events may are triggered in order: OnSelectedRowsChanged, OnSelectedColsChanged, OnSelectionChanged. You can

check ColsChanged flag to determine the OnSelectedColsChanged event will be triggered next. {* ver. 4.3 *}

See also: TXCustomDBGrid.OnSelectionChanged, TXCustomDBGrid.OnSelectedColsChanged, TXCustomDBGrid.ColsChanged, , TXCustomDBGrid.RowsChanged, TXCustomDBGrid.PerformDelayed

TXCustomDBGrid.OnSelectionChanged

Occurs when the selected rows or columns changed.

```
property OnSelectionChanged: TNotifyEvent; {* ver. 4.3 *}
```

Description
Write an OnSelectionChanged event handler to take specific action when SelectedRows and/or SelectedCols have been changed in the grid. This event is triggered after group of changes were made is SelectedRows or SelectedCols (eg. when SelectAll was performed the event is triggered once - after all changes in SelectedRows and/or SelectedCols list. Use BeginSelect & EndSelect statements when group of changes is made in SelectedCols and/or SelectedRows list directly in your code. To minimize numbers of calls for this event the dgDelaySelectedRows option should be included in OptionsEx. This option is common for events: OnSelectedRowsChanged, OnSelectedColsChanged, OnSelectionChanged.

Note. In some cases (eg. when MultiSelect is msCells) after one group of changes the events may are triggered in order: OnSelectedRowsChanged, OnSelectedColsChanged, OnSelectionChanged. You can check RowsChanged and ColsChanged flag to determine the OnSelectedRowsChanged and OnSelectedColsChanged events were triggered prior. You should use this event when your code is independent of kind of current selection (eg. CopyToClipboard).

See also: TXCustomDBGrid.OnSelectedColsChanged, TXCustomDBGrid.OnSelectedRowsChanged, TXCustomDBGrid.ColsChanged, , TXCustomDBGrid.RowsChanged, TXCustomDBGrid.PerformDelayed

TXCustomDBGrid.OnSequenceUpdated

Occurs when the sequence numbers have been recalculated.

```
property OnSequenceUpdated: TNotifyEvent;
```

Description
Write an OnSequenceUpdated event handler to take specific action when sequence numbers (RecNumber, RecCount) have been recalculated. Sequence numbers must be recalculated when ForcedSequence is True. OnSequenceUpdated is triggered when UpdateSequence finished calculations and you can use sequence numbers for further actions.

See also: TXCustomDBGrid.UpdateSequence

TXCustomDBGrid.OnToolMenuClick

Occurs when the user clicks any menu item from ToolMenu.

```
property OnToolMenuClick: TNotifyEvent; {* ver. 8.0 *}
```

Description
Write an OnToolMenuClick event handler to take specific action when the user clicks any or specific menu item from ToolMenu. The Sender parameter is the menu item clicks by the user.

See also: TCustomDBGrid.OnToolMenuCreate, TCustomDBGrid.OnToolMenuUpdate, TCustomDBGrid.OnToolMenuPopup

TXCustomDBGrid.OnToolMenuCreate

Occurs when the ToolMenu creates menu items.

```
property OnToolMenuCreate: TNotifyEvent; {* ver. 8.0 *}
```

Description
Write an OnToolMenuCreate event handler to add your custom menu items to the ToolMenu. The Sender parameter is the created tool menu.

See also: TCustomDBGrid.OnToolMenuUpdate, TCustomDBGrid.OnToolMenuPopup, TCustomDBGrid.OnToolMenuClick

TXCustomDBGrid.OnToolMenuPopup

Occurs when the ToolMenu is popped up on the screen.

```
property OnToolMenuPopup: TNotifyEvent; {* ver. 8.0 *}
```

Description
Write an OnToolMenuPopup event handler to update state of any menu item in the ToolMenu (property Enabled or Checked). The Sender parameter is the tool menu.

See also: TCustomDBGrid.OnToolMenuCreate, TCustomDBGrid.OnToolMenuUpdate, TCustomDBGrid.OnToolMenuClick

TXCustomDBGrid.OnToolMenuUpdate

Occurs when the ToolMenu updates visibility of menu items.

```
property OnToolMenuUpadate: TNotifyEvent; {* ver. 8.0 *}
```

Description
Write an OnToolMenuUpdate event handler to hide or show any menu item in the ToolMenu (property Visible). The Sender parameter is the tool menu.

See also: TCustomDBGrid.OnToolMenuCreate, TCustomDBGrid.OnToolMenuPopup, TCustomDBGrid.OnToolMenuClick

TXCustomDBGrid.OnTotalCalcQuery

Occurs for each record when the grid recalculates total value.

```
type
  TTotalCalcQueryEvent = procedure (Sender: TObject; DataSet: TDataSet; var
    Accept: Boolean) of object;
property OnTotalCalcQuery: TTotalCalcQueryEvent;
```

Description
Write an OnTotalCalcQuery event handler to take specific action when the grid recalculates total value. A total value appears in total cell when TotalResult is trCalcValue. OnTotalCalcQuery is triggered for each record recalculated by CalcValue's when UpdateTotals is performed. TXDBGrid object is passed as Sender.

The DataSet parameter determines recalculated record (current record in DataSet). The Accept parameter is default True. You can set Accept to False if you want to exclude current record from calculation for all CalcValue functions defined in any total cell. If Accept will be changed to False, OnTotalCalcValue event will be omitted for this record for all total cells. It allows to apply a global filter for calculated totals values.

See also: TXCustomDBGrid.OnTotalCalcValue, TXCustomDBGrid.OnTotalUpdated, TXCustomDBGrid.UpdateTotals

TXCustomDBGrid.OnTotalCalcValue

Occurs for each total cell when the grid recalculates total value.

```
type
  TCalcPass = (cpReset, cpCalcRec, cpFinal);
  TTotalCalcValueEvent = procedure (Total: TXColumnTotalValue; Pass:
    TCalcPass; var Value: Variant) of object;
property OnTotalCalcValue: TTotalCalcValueEvent;
```

Description
Write an OnTotalCalcValue event handler to take specific action when the grid recalculates total value. A total value appears in total cell when TotalResult is trCalcValue. OnTotalCalcValue is triggered for all CalcValue's when UpdateTotals is performed.

The Total parameter determines the total cell where the result of calculation will be displayed. The Value parameter contains the value that will be processed by CalcValue function. You can modify this value to modify the result of calculation. The Pass parameter determines the pass of calculation. These are the possible values of Pass:

Value	Meaning
cpReset	The Value parameter contains the initial value for CalcValue function. Default initial Value is Unassigned.
cpCalcRec	The Value parameter contains Field.Value for current record. This pass is repeated for each processed record in dataset. You can modify this Value to modify result of calculation or set Value to Null to exclude selected records. See also NullValue property. The temporary result of calculation is stored in TempValue public property. The count of processed values is stored in Counter public property. You can use those public property in this event, but you should'nt directly modify Value property.
cpFinal	The Value parameter contains the final result of calculation. You can modify this Value to modify final result before it appears in total cell.

You can write OnTotalCalcValue event handler to modify result of standard CalcValue function, but in

particular you should write this event when CalcValue is cvCustom. Then, you can calculate here any custom function.

OnTotalCalcValue is frequently triggered. To reduce the frequency with which OnTotalCalcValue occurs, set dgDelayUpdateTotals in OptionsEx or set CalcValues property to false when the calculation is not needed. Remember, the code in this event handler should be simple.

To exclude some records for all CalcValue's in the grid (global calculation filter) you should write OnTotalCalcQuery event handler.

See also: TXCustomDBGrid.OnTotalCalcQuery, TXCustomDBGrid.OnTotalsUpdated, TXCustomDBGrid.UpdateTotals

TXCustomDBGrid.OnTotalClick

Occurs when the user releases the mouse in one of the total cells.

```
type
  TTotalClickEvent = procedure (Column: TXColumn; Index: Integer) of object;
property OnTotalClick: TTotalClickEvent;
```

Description
Write an OnTotalClick event handler to take specific action when the user clicks in one of the total cells. The Column parameter is the TXColumn object that corresponds to the column where the mouse was when the user released the left mouse button. The Index parameter corresponds to the number of total row. Use Index property as index of TotalRows or TotalValueRows property to obtain the total cell when the user released the left mouse button.

See also: TCustomDBGrid.OnTitleClick

TXCustomDBGrid.OnTotalGetText

Occurs when the grid retrieves Text for total cell.

```
type
  TTotalGetTextEvent = procedure (Column: TXColumn; Index: Integer; var Text:
    string; TextKind: TTextKind) of object;
property OnTotalGetText: TTotalGetTextEvent;
```

Description
Write an OnTotalGetText event handler to change Text retrieved for total cell. OnTotalGetText event is called by TotalGetText protected method when the grid retrieves Text to appear it within total cell. Value of Text parameter depends on current value of TotalResult property and other TXColumnTotal properties.

The Column parameter determines column of total cell, the Index parameter corresponds to the number of total row. The TextKind parameter is reserved for future use (tkDisplay by default). You can use Index property as index of TotalRows or TotalValueRows property to determine the total cell for which the Text is retrived.

See also: TXCustomDBGrid.GetTotalText

TXCustomDBGrid.OnTotalsUpdated

Occurs when the total values have been recalculated.

```
property OnTotalsUpdated: TNotifyEvent;
```

Description
Write an OnTotalsUpdated event handler to take specific action when all total values have been recalculated. A total value appears in total cell when TotalResult is trCalcValue. OnTotalsUpdated is triggered when UpdateTotals finished calculations and all CalcValue's are recalculated. Then, you can use results for further actions.

See also: TXCustomDBGrid.UpdateTotals, TXCustomDBGrid.OnTotalCalcValue, , TXCustomDBGrid.OnTotalCalcQuery

TXDBGridColumns

TXDBGridColumns represents a container for TXColumn objects.

Unit
XDBGrids

Description
Each TXDBGridColumns holds a collection of TXColumn objects in a data grid (TXDBGrid). TXDBGridColumns maintains an index of the columns in its Items array. The Count property contains the number of columns in the collection. At design time, use the data grid's Columns editor to add, remove, or modify columns.

See also: TDBGridColumns, TXColumn, TXCustomDBGrid.Columns, TXDBGrid

TXDBGridColumns.Grid

Indicates the XDBGrid control to which the TXDBGridColumns instance belongs.

```
property Grid: TXCustomDBGrid;
```

Description
The read-only Grid property indicates which XDBGrid control owns the TXDBGridColumns object.

See also: TXCustomDBGrid.Columns

TXDBGridColumns.Items

Lists the columns in the collection.

```
property Items[Index: Integer]: TXColumn default;
```

Description
Use Items to access individual columns. The value of the Index parameter corresponds to the Index property of TXColumn. It represents the position of the column in the XDBGrid.

Items is the default property of TXDBGridColumns. This means that the property name, Items, can be omitted when referring to columns of the TXDBGridColumns object. Thus, the line

```
FirstCol := XDBGrid1.Columns.Items[0];
```

can also be written

```
FirstCol := XDBGrid1.Columns[0];
```

See also: TCollection.Index

TXDBGridColumns.Settings

Contains lines of text for column layout settings.

```
property Settings: TStrings; {* ver. 8.0 *}
```

Description
Settings is a TStrings object designed for store and retrieve column layout settings as name-value pairs. For more information on name-value pairs, refer to Values property.

The Settings object contains several lines with base settings (FieldName, Width, Visible, Expanded, Report.Visible) for each column. Name of each line determine current position of the column in the XDBGrid (e.g. "Column1", "Column2", ...). The line with name "Columns" contains number of columns in the grid.

When soStretchMode is included in Settings.Options, the additional line with name "StretchMode" is added to Settings object. Starting from RAD Studio 10 Seattle additional line named "CurrentPPI" is also always added. This line allows to scale column widths for different screen resolutions.

This convention corresponds to the format of configuration files (*.ini). For example (gray lines are optional):

Columns=4
Column1=ID,20,0,0,1,1
Column2=FirstName,40,1,0,1,1
Column3=LastName,40,1,0,1,1
Column4=Country,30,1,0,1,1
CurrentPPI=96
StretchMode=0

The Settings object is designed to store and retrieve column layout to/from ini file or to/from registry but you can also use TStrings methods like: LoadFromFile/SaveToFile, LoadFromStream/SaveToStream, GetTextStr/SetTextStr (or Text property) to load/save column Settings to text file, any stream (BLOB field) or memory variable. Each time you are reading Settings property the current column settings are retrieved from XDBGrid's properties. Each time you are setting new Settings property the XDBGrid's columns properties are modified. After you load Settings.Text property from the text file, stream or variable you need to call ApplySettings method to apply changes to the grid columns.

The contains of Settings property is main part of Settings.Layout property.

See also: TXCustomDBGrid.Columns, TXDBGridColumns.ApplySettings

TXDBGridColumns.Add

Creates a new TXColumn instance and adds it to the Items array.

```
function Add: TXColumn;
```

Description
Add returns the new column. At design time use the XDBGrid's Columns editor to add columns to the grid.

See also: TXCustomDBGrid.Columns, TXDBGridColumns.Items

TXDBGridColumns.AddAutoExpand

Creates a new auto-expanded TXColumn instance and adds it to the Items array.

```
function AddAutoExpand: TXColumn; {* ver. 6.6 *}
```

Description
AddAutoExpand returns the new column with empty FieldName and ExpandBox = True. See also: DefAutoExpandTitle, DefAutoExpandWidth global variable.

See also: TXCustomDBGrid.Columns, TXDBGridColumns.Add, TXCustomDBGrid.OptionsExt

TXDBGridColumns.AddAutoNumber

Creates a new auto-numbered TXColumn instance and adds it to the Items array.

```
function AddAutoNumber: TXColumn; {* ver. 4.3 *}
```

Description
AddAutoNumer returns the new column with empty FieldName and AutoNumber = True. See also: DefAutoNumberTitle, DefAutoNumberWidth global variable.

See also: TXCustomDBGrid.Columns, TXDBGridColumns.Add, TXCustomDBGrid.OptionsExt

TXDBGridColumns.AddAutoSelect

Creates a new auto-selected TXColumn instance and adds it to the Items array.

```
function AddAutoSelect: TXColumn; {* ver. 4.3 *}
```

Description
AddAutoSelect returns the new column with empty FieldName and CheckBox = True. See also: DefAutoSelectTitle, DefAutoSelectWidth global variable.

See also: TXCustomDBGrid.Columns, TXDBGridColumns.Add, TXCustomDBGrid.OptionsExt

TXDBGridColumns.ApplySettings

Applies changes in column layout stored in Settings buffer.

```
function ApplySettings; {* ver. 8.0 *}
```

Description
Use ApplySettings to apply new columns Settings in the grid. This procedure is useful after you load new column layout to Settings buffer by calling LoadFromFile, LoadFromStream methods or after you setting the new value for property Text.

See also: TXCustomDBGrid.Columns, TXDBGridColumns.Settings

TXDBGridColumns.Create

Creates and initializes a TXDBGridColumns object.

```
type
  TXColumnClass = class of TXColumn;
constructor Create(Grid: TXCustomDBGrid; ColumnClass: TXColumnClass);
```

Description
The Create method takes two parameters: a XDBGrid instance object and TXColumn (or the name of a class derived from TXColumn).

See also: TXCustomDBGrid.Columns, TXDBGrid

TXDBGridColumns.GetEnumerator

Creates enumerator for TXDBGridColumns class.

```
function GetEnumerator: TXDBGridColumnsEnumerator; {* ver. 5.6 *} // For
Delphi 2009 or higher
```

Description

You not need to call this function directly. It allows you to iterate over all columns in the grid.

```
type
  Column: TXColumn;
begin
  for Column in XDBGrid1.Columns do
  begin
    // Perform action on each column in XDBGrid1.Columns
  end;
end;
```

See also: TXCustomDBGrid.DataRows, TXColumnList.GetEnumerator

TXDBGridColumns.RemoveMarkers

Removes all sorting markers from titles.

```
procedure RemoveMarkers;
```

Description

Use the RemoveMarkers method to remove all markers from titles.

See also: TXDBGridColumns.UpdateMarkers

TXDBGridColumns.UpdateMarkers

Updates all sorting markers in titles.

```
procedure UpdateMarkers;
```

Description

Use UpdateMarkers when you made any special changes you need call this method.

See also: TXDBGridColumns.RemoveMarkers

TXColumn

TXColumn represents a column in a data grid (TXDBGrid).

Unit
XDBGrids

Description
Each TXDBGrid uses a TXDBGridColumns to maintain a collection of TXColumn objects. Each TXColumn object represents the visual characteristics and data binding of a column in the grid.

See also: TColumn, TCollection, TCollectionItem, TXDBGridColumns, TXDBGrid

TXColumn.AutoNumber

Specifies whether the column is autonumbered column.

```
property AutoNumber: Boolean;
```

Description
When AutoNumber is True the column display RecNumber's value instead FieldName's value. You may also define additional column with empty FieldName property and set AutoNumber for that.

The AutoNumber values are formatted by AutoNumberStr function. The developer can change DefAutoNumberFormat or formatted text in OnColGetText event handler. When AutoNumber is True the column can't be edited. Usually the column with AutoNumber is placed in FixedCols range and the property Alignment has set to taRightJustify.

You should use AutoNumber property only when ForcedSequence or IsSequenced is True.

See also: TXCustomDBGrid.RecNumber, TXCustomDBGrid.ForcedSequence, AutoNumberStr

TXColumn.ButtonStyle

Determines whether and how the user can select values for the column from a list, menu or other drop-down editor.

```
type
  TXColumnButtonStyle = (cbsAuto, cbsEllipsis, cbsNone, cbsDropDown,
    cbsDropDownMenu, cbsCalendar, cbsCalculator);
property ButtonStyle: TXColumnButtonStyle;
```

Description
The ButtonStyle property determines how users can edit the data of a data-grid column. These are the possible values of ButtonStyle:

Value	Meaning
cbsAuto	If the column is associated with a lookup field or has a value assigned to its PickList property, the grid displays a combo box in the column. The user can choose a value from the drop-down list. If the column is associated with TDateField or TDateTimeField the user can choose date value from the drop-down calendar. Time part of TDateTime value leaves unchanged. See also: cbsCalendar.

cbsEllipsis	The column displays an ellipsis button that the user can click to choose a value. Clicking the ellipsis button triggers an OnEditButtonClick event.
cbsNone	No combo box or ellipsis button is provided. The user cannot select the column's content from a list.
cbsDropDown	The column displays a drop-down button that the user can click to choose a value. Clicking the drop-down button triggers an OnEditButtonClick event.
cbsDropDownMenu	The column displays a drop-down button that the user can click to choose a value from drop-down menu. Clicking the drop-down button DropDownMenu is automatically dropped down.
cbsCalendar	The column displays a drop-down button that the user can click to choose a date from drop-down calendar. Clicking the drop-down button the month calendar is automatically dropped down. The calendar requires date value compatible with ShortDateFormat global variable. You can also use cbsCalendar value when any text field contains date or datetime value.
cbsCalculator	The column displays a drop-down button that the user can click to calculate input value in drop-down calculator. Clicking the drop-down button the comfortable calculator is automatically dropped down. The calculator is similar in use to Microsoft Calculator.

See also: TXColumn.ExpandStyle, TXColumn.PickList, TCustomDBGrid.OnEditButtonClick, TXColumn.DropDownMenu, TXColumn.ListOptions

TXColumn.CalendarActive

Specifies whether the column has a drop-down calendar.

```
property CalendarActive: Boolean; {* ver. 6.0 *}
```

Description
Use CalendarActive property to determine the column has defined drop-down calendar.

CalendarActive is a read-only property.

See also: TXColumn.ButtonStyle, TXColumn.LookupActive, TXColumn.PickDataActive

TXColumn.CalendarDepth

Specifies maximum depth of drop-down calendar in the column.

```
type
  TCalendarDepth = (cdMonth, cdYear, cdDecade, cdCentury);
property CalendarDepth: TCalendarDepth; {* ver. 6.5 *}
```

Description
Use CalendarDepth property to select maximum depth of drop-down calendar in the column. These are the possible values of CalendarDepth:

Value	Meaning
cdMonth	The drop-down calendar shows only months.
cdYear	The drop-down calendar shows only months and years.
cdDecade	The drop-down calendar shows only months, years and decades.
cdCentury	The drop-down calendar shows months, years, decades and centuries.

See also: TXColumn.ButtonStyle, TXColumn.CalendarActive

TXColumn.CharCase

Determines the case of the text within the inplace editor.

```
property CharCase: TEditCharCase; {* ver. 4.31 *}
```

Description
Use CharCase to force the contents of the inplace editor to assume a particular case. The possible values of CharCase are as follows:

Value	Meaning
ecLowerCase	The text is converted to lowercase
ecNormal	The text appears in mixed case. It is not forced into any case.
ecUpperCase	The text is converted to uppercase.

When CharCase is set to ecLowerCase or ecUpperCase, the case of characters is converted as the user types them into the inplace editor. Changing the CharCase property to ecLowerCase or ecUpperCase changes the actual contents of the text, not just the appearance. Any case information is lost and can't be recaptured by changing CharCase to ecNormal.

See also: TXColumn.WordWrap

TXColumn.CheckBox

Specifies whether the column has check boxes.

```
property CheckBox: Boolean;
```

Description
Set CheckBox to True to make check boxes appear in the column. Each row in the grid displays a single check box.

The CheckBox property may be used for any field type that can return value in AsString property. When FieldName property is empty for the column, state of check boxes correspond to bookmarks collected in SelectedRows.

See also: TXColumn.CheckBoxThemed, TXColumn.CheckBoxToggle, TXColumn.CheckBoxValues, TXColumn.IsCheckBoxValue, TXColumn.ToggleCheckBoxValue, TXColumn.CheckBoxStyle, TXColumn.CheckBoxKind

TXColumn.CheckBoxKind

Specifies kind of sign in the check box.

```
type
  TCheckBoxKind = (ckCheck, ckCross, ckBox);
property CheckBoxKind: TCheckBoxKind;
```

Description
Use CheckBoxKind to select a kind of sign drawn in the check box. The CheckBoxKind property is practicable for non-Windows XP styles or either when Images property defines customized checkboxes for the column. These are the possible values of CheckBoxKind:

Value	Meaning
ckCheck	The check box sign is a check sign
ckCross	The check box sign is a cross sign
ckBox	The check box sign is a box sign

See also: TXColumn.CheckBoxTXColumn.CheckBoxThemed, , TXColumn.CheckBoxValues, TXColumn.IsCheckBoxValue, TXColumn.ToggleCheckBoxValue, TXColumn.CheckBoxStyle, TXColumn.CheckBoxToggle

TXColumn.CheckBoxStyle

Specifies style of check box drawn in the column.

```
type
  TCheckBoxStyle = (cbAuto, cbAutoGray, cbAutoLowered, cbDefault, cbSoft,
    cbLoweredDark, cbLoweredGray, cbLoweredMild, cbFlat, cbGray, cbMild);
property CheckBoxStyle: TCheckBoxStyle;
```

Description
Use CheckBoxStyle to select a style of check box drawn in the column. The CheckBoxStyle property is practicable only for non-Windows XP styles. These are the possible values of CheckBoxStyle:

Value	Meaning
cbAuto	The style of check box is dependent on current FixedStylevalue.
cbAutoGray	The style of check box is one from Gray styles dependent on current FixedStyle value.
cbAutoLowered	The style of check box is one from Lowered styles dependent on current FixedStyle value.
cbDefault	The style of check box is 3D - suitable for FixedStyle = fsDefault.
cbSoft	The style of check box is 3D - suitable for FixedStyle = fsSoft.
cbLoweredDark	The style of check box is concave - suitable for FixedStyle = fsNice.
cbLoweredGray	The style of check box is concave - suitable for all FixedStyle values.
cbLoweredMild	The style of check box is concave - suitable for FixedStyle = fsFine.
cbFlat	The style of check box is flat - suitable for FixedStyle = fsFlat.
cbGray	The style of check box is flat - suitable for all FixedStyle values.
cbMild	The style of check box is flat - suitable for FixedStyle = fsMild.

See also: TXColumn.CheckBox, TXColumn.CheckBoxThemed, TXColumn.CheckBoxValues, TXColumn.IsCheckBoxValue, TXColumn.ToggleCheckBoxValue, TXColumn.CheckBoxKind, TXColumn.CheckBoxToggle

TXColumn.CheckBoxThemed

Specifies whether the column has themed check boxes.

```
property CheckBoxThemed: Boolean; {* ver. 5.0 *}
```

Description
Set CheckBoxThemed to True to force themed check boxes in the column.

_____
See also: TXColumn.CheckBox, TXColumn.CheckBoxToggle, TXColumn.CheckBoxValues, TXColumn.IsCheckBoxValue, TXColumn.ToggleCheckBoxValue, TXColumn.CheckBoxStyle, TXColumn.CheckBoxKind

---

TXColumn.CheckBoxToggle

Controls how the state of check box may be toggled by the user.

```
type
  TCheckBoxToggle = (ctCustomToggle, ctKeyPressOnly, ctOnCellClick,
    ctSelectClick);
property CheckBoxToggle: TCheckBoxToggle;
```

Description
Set the CheckBoxToggle property to determine how the user may toggle state of check box. These are the possible values of CheckBoxToggle:

Value	Meaning
ctCustomToggle	The check box is'nt toggled by the XDBGrid - use custom toggle
ctKeyPressOnly	The check box is toggled when the user presses Enter or Space
ctOnCellClick	The check box is toggled when the user presses Enter or Space or clicks on the cell
ctSelectClick	The check box is toggled when the user presses Enter or Space or clicks on the selected cell

_____
See also: TXColumn.CheckBox, TXColumn.CheckBoxThemed, TXColumn.CheckBoxValues, TXColumn.IsCheckBoxValue, TXColumn.ToggleCheckBoxValue, TXColumn.CheckBoxStyle, TXColumn.CheckBoxKind

TXColumn.CheckBoxValues

Controls how the field's value is translated to checkbox state.

```
property CheckBoxValues: string;
```

Description
Use CheckBoxValues to specify strings you want used to represent checkbox state. Use any pair of phrases, separated by a semicolon. Default phrases for CheckBoxValues are 'True;False' that corresponds to values of Boolean field getting by AsString property. The CheckBox property may be used for any field type that can return value in AsString property.

The string associated with checkbox state can be an empty string. To set the value for True to an empty string, set CheckBoxValues to a string that begins with a semicolon. To associate False with an empty string, set CheckBoxValues to the string for True, with no semicolon at all. For example, to associate True with the string Print, and False with an empty string, set CheckBoxValues to 'Print'.

These strings are also used to setting field's value when checkbox is toggled.

See also: TXColumn.CheckBox, TXColumn.CheckBoxToggle, TXColumn.IsCheckBoxValue, TXColumn.ToggleCheckBoxValue, TXColumn.CheckBoxStyle, TXColumn.CheckBoxKind

TXColumn.DelimiterChar

Indicates when multi selected values are delimited by specified character.

```
property DelimiterChar: Char; {* ver. 6.7 *}
```

Description
When the column has associated MultiSelectList and property DelimiterLine is False, the multi selected values are delimited by character specified in DelimiterChar property when DelimiterText is empty.

See also: TXColumn.DelimiterLine, TXColumn.DelimiterText, TXColumn.MultiSelectList

TXColumn.DelimiterLine

Indicates when multi selected values are delimited by new line.

```
property DelimiterLine: Boolean; {* ver. 6.7 *}
```

Description
When MultiSelectListis associated with the column and DelimiterLine property is True then multi selected values are delimited by new line (CR LF). When DelimiterLine is False, multi selected values are delimited by DelimiterChar or DelimiterText.

See also: TXColumn.DelimiterChar, TXColumn.DelimiterText, TXColumn.MultiSelectList

TXColumn.DelimiterText

Indicates when multi selected values are delimited by few fixed characters.

```
property DelimiterText: string; {* ver. 8.0 *}
```

Description
When the column has associated MultiSelectList and property DelimiterLine is False, the multi selected values can be delimited by few fixed characters specified in DelimiterText property. Remember, when DelimiterText is not empty the DelimiterChar is also used internally.

See also: TXColumn.DelimiterChar, TXColumn.MultiSelectList

TXColumn.DialogOptions

Specifies additional column's options for TXDBColumnsDialog component.

```
type
  TDialogOption = (doShowColumn, doItemEnabled);
  TDialogOptions = set of TDialogOption;
property DialogOptions: TDialogOptions;
```

Description
Set DialogOptions to include additional options for TXDBColumnsDialog component. These are the possible values of DialogOptions:

Value	Meaning
doShowColumn	The column appears on TXDBColumnsDialog's list.
doItemEnabled	TXDBColumnsDialog's list item is enabled and Visible property can be changed in dialog.

See also: TXDBColumnsDialog, TXColumnReport.DialogOptions

TXColumn.DisplayText

Represents the column's value as it is displayed in the grid.

```
property DisplayText: string;
```

Description
DisplayText is a read-only string representation of a column's value for displaying in the grid. It represents the column's value when it is not being edited. When the cell is being edited, the EditText value is retrieved from dataset.

If the grid has an OnColGetText event handler, DisplayText is the value returned in the Text parameter of the OnColGetText event handler when TextKind parameter is tkDisplay. Otherwise, DisplayText is the value of the field's DisplayText property. See also ShowBlob and AutoNumber property.

See also: TXColumn.EditText

TXColumn.DropDownMenu

Identifies the drop-down menu associated with the column.

```
property DropDownMenu: TPopupMenu;
```

Description
Assign a value to DropDownMenu to make a drop-down menu appear when the user clicks the edit button of column. If the ButtonStyle property is cbsAuto when DropDownMenu is assigned, the ButtonStyle property is automatically changed to cbsDropDownMenu and the edit button will be appear in the edited cell of column. If the ButtonStyle is cbsDropDownMenu when DropDownMenu is removed, the ButtonStyle property is automatically changed to cbsAuto. See also: DropDownPoint.

See also: TColumn.PopupMenu, TXColumnTitle.DropDownMenu, TXColumn.ButtonStyle, TXCustomDBGrid.FillerPopupMenu, TXCustomDBGrid.IndicatorPopupMenu

TXColumn.DropDownRows

Specifies the number of lines displayed in the column's drop-down list.

```
property DropDownRows: Integer;
```

Description
DropDownRows determines the number of lines of text displayed in the drop-down list associated with the column. This property is used only if ButtonStyle is set to cbsAuto and the column has a lookup field or pick list associated with it.

When DropDownRows property is 0 (default) and drop-down list has more then MaxDropDownRows items, then MaxDropDownRows items is displayed in the drop-down list, otherwise all list items are displayed. {* ver. 4.3 *}

See also: TXColumn.DropDownWidth, TXColumn.PickList, MaxDropDownRows

TXColumn.DropDownWidth

Specifies a width of the column's drop-down list.

```
property DropDownWidth: Integer;
```

Description
DropDownWidth determines a width of the drop-down list associated with the column. This property is used only if ButtonStyle is set to cbsAuto and the column has a lookup field or pick list associated with it. If DropDownWidth value is less then column's width the column's width will be used as the width of the drop-down list.

When DropDownWidth property is 0 (default) and auto-calculated width of drop-down list is greater then MaxDropDownWidth, then displayed width of drop-down list is limited to MaxDropDownWidth. {* ver. 4.3 *}

See also: TXColumn.DropDownRows, TXColumn.PickList, MaxDropDownWidth

TXColumn.EditMemo

Determines whether the memo field may be automatically edited in an inplace editor.

```
property EditMemo: Boolean;
```

Description
If EditMemo is set to False (default), memo field cannot be edited in the grid by the user unless OnGetText/OnSetText event handlers are defined for this field. If EditMemo is set to True, memo field can be edited in the grid without this events. Remember, there is'nt enough place to edit multiline memo fields in the inplace editor. You should set this property to True only for single line memo. To edit memo field in the grid the ShowEdit property must be set to True. See also EditText property.

_____
See also: TXColumn.ShowEdit

TXColumn.EditorHint

Specifies the text string that can appear when the user moves the mouse pointer over a edited cell.

```
property EditorHint: string;
```

Description
Set EditorHint to a string that provides information about the editing of the field. The hint text appears in a Help Hint window when the user pauses with the mouse over the edited cell if HintOptions includes hoShowEditorHints.

_____
See also: TXCustomDBGrid.HintOptions, TXCustomDBGrid.OnCellHint, TXCustomDBGrid.FillerHint, TXCustomDBGrid.IndicatorHint, TXColumn.Hint, TXColumnTitle.Hint

TXColumn.EditText

Contains the string to display in an inplace editor when the grid is in edit mode.

```
property EditText: string;
```

Description
The inplace editor rely on the EditText property to provide the editing format for each column. For example, by default the EditText property of a currency field omits the thousands separators and currency symbol.

EditText can differ from the DisplayText property if the column uses a different string representation when the value is being edited. To implement two different string representations of a column's value, use the OnColGetText event handler. If an OnColGetText event handler is assigned, EditText is the value returned in the Text parameter of the event handler when the TextKind parameter is tkEdit. If there is no OnColGetText event handler, EditText is the value of the field's Text property. See also OnColSetText, EditMemo and AutoNumber property.

_____
See also: TXColumn.DisplayText, TXColumn.EditMemo

TXColumn.Ellipsis

Specifies whether the data is completed by ... when it is too long for the width of the column.

```
property Ellipsis: Boolean;
```

Description
Set Ellipsis to True to allow the data completed by triple dot. When Ellipsis is True, text that is too wide for the column is completed by triple dot.

When Ellipsis is False, text that is too wide for the column appears truncated. See also WordWrap property.

See also: TXColumn.WordWrap, TXColumnTotal.Ellipsis, TXColumnTitle.Ellipsis

TXColumn.ExpandBox

Specifies whether the column has expand boxes.

```
property ExpandBox: Boolean; {* ver. 6.6 *}
```

Description
Set ExpandBox to True to make expand boxes appear in the column. Each row in the grid displays a single expand box.

The ExpandBox property may be used for any field type and also when FieldName property is empty for the column.

Look at the example in OnCellExpand topic, how to expand additional grid for the cell.

See also: TXColumn.ExpandBoxThemed, TXColumn.ExpandBoxToggle, TXColumn.ExpandBoxSize

TXColumn.ExpandBoxSize

Specifies a size of sign in the expand box.

```
type
  TExpandBoxSize = (ebSmall, ebMedium, ebLarge);
property ExpandBoxSize: TExpandBoxSize; {* ver. 6.6 *}
```

Description
Use ExpandBoxSize to select a size of sign drawn in the expand box. The ExpandBoxSize property is useful only when Themes and Style are not used. These are the possible values of ExpandBoxSize:

Value	Meaning
ebSmall	The expand box sign has a small size
ebMedium	The expandbox sign has a medium size
ebLarge	The expandbox sign has a large size

See also: TXColumn.ExpandBoxThemed, TXColumn.ExpandBoxToggle, TXColumn.ExpandBox

TXColumn.ExpandBoxThemed

Specifies whether the column has themed expand boxes.

```
property ExpandBoxThemed: Boolean; {* ver. 6.6 *}
```

Description
Set ExpandBoxThemed to True to force themed expand boxes in the column.

See also: TXColumn.ExpandBox, TXColumn.ExpandBoxToggle, TXColumn.ExpandBoxSize

TXColumn.ExpandBoxToggle

Controls how the state of expand box may be toggled by the user.

```
type
  TExpandBoxToggle = (etCustomToggle, etKeyPressOnly, etOnCellClick,
    etSelectClick);
property ExpandBoxToggle: TExpandBoxToggle; {* ver. 6.6 *}
```

Description
Set the ExpandBoxToggle property to determine how the user may toggle state of expand box. These are the possible values of ExpandBoxToggle:

Value	Meaning
etCustomToggle	The expand box is'nt toggled by the XDBGrid - use custom toggle
etKeyPressOnly	The expand box is toggled when the user presses Enter or Space
etOnCellClick	The expand box is toggled when the user presses Enter or Space or clicks on the cell
etSelectClick	The expand box is toggled when the user presses Enter or Space or clicks on the selected cell

See also: TXColumn.ExpandBoxThemed, TXColumn.ExpandBox, TXColumn.ExpandBoxSize

TXColumn.ExpandBoxTree

Specifies whether the column has tree view.

```
property ExpandBoxTree: Boolean; {* ver. 7.0 *}
```

Description
Set ExpandBoxTree to True to make tree view appear in the column. Each row in the grid displays a single expand box for tree view.

The ExpandBoxTree property may be used for any field type and also when FieldName property is empty for the column.

When ExpandBoxTree is True the property ExpandBox is also True and the ImageMargin must be greather then 0.

When TreeView.ColumnNames is not empty the ExpandBoxTree property is controlled automaticaly for selected ColumnNames.

See also: TXColumn.ExpandBox, TXColumn.ExpandBoxSize, TXColumn.ImageMargin

TXColumn.Expandable

Specifies whether the column can be expanded to show any child columns.

```
property Expandable: Boolean;
```

Description
Expandable will be False if the represented field is not an ADT, array, or reference field and ExpandCols property is 0.

See also: TXColumn.ExpandWidth, TXColumn.ExpandWidth, TXColumn.Expanded, TXColumn.ExpandCols, TXColumn.ExpandStyle, TADTField

TXColumn.ExpandCols

Specifies the number of following columns on the right side of the column that are child columns.

```
property ExpandCols: Integer;
```

Description
Set ExpandCols for ANY column to "convert on the fly" the following columns to child columns (in TDBGrid component the column has child columns only if the represented field is an ADT, array, or reference field). If ExpandCols property is'nt 0, the column is also Expandable and the child columns can be Expanded and contracted.

To create Expandable column you can use one of existing column or define calculated column (recommended) and add it to the grid (see Demo1 example) or define empty grid column (DataField=nil) and create OnColGetText event for this.

See also: TXColumn.ExpandWidth, TXColumn.ExpandStyle, TXColumn.ExpandIndex, TXColumn.Expanded, TXColumn.Expandable

TXColumn.Expanded

Specifies whether the column is expanded.

```
property Expanded: Boolean;
```

Description
Expanded only applies to Expandable columns. When a column is expanded, a column appears for each child field of the object field or for each child column of ParentColumn when ExpandCols is'nt 0. These child columns each have a column title and appear under the column title of the parent column. When Expanded is False, child field values are summarized in a comma-delimited string in the object field's column (TADTField) or in the OnCalcFields event handler, and cannot be edited.

Expandable columns can also be expanded and contracted at runtime through the UI by clicking on the expand button that appears in the column title when ExpandStyle property is cesAuto (default).

See also: TXColumn.ExpandWidth, TXColumn.ExpandIndex, TXColumn.Expandable, TXColumn.ExpandCols, TXColumn.ExpandStyle, TADTField

TXColumn.ExpandIndex

Indicates the position of the last expanded column.

```
property ExpandIndex: Integer;
```

Description

When column is Expandable, ExpandIndex indicates the position of the last Expanded column.

See also: TXColumn.ExpandWidth, TXColumn.ExpandStyle, TXColumn.ExpandCols, TXColumn.Expanded, TXColumn.Expandable, TADTField

TXColumn.ExpandStyle

Determines whether and how the user can use expand button in column title.

```
type
  TXColumnExpandStyle = (cesAuto, cesNone, cesDropDown, cesDropDownMenu,
    cesFilterList, cesFilterForm); {* ver. 6.0 *} {* ver. 7.1 *}
property ExpandStyle: TXColumnExpandStyle;
```

Description

The ExpandStyle property determines how users can use expand button in column title. These are the possible values of ExpandStyle:

Value	Meaning
cesAuto	If the column is Expandable, the grid displays an expand button in the column title. The user can click expand button to expand/contract expandable column. See also: Expanded.
cesNone	No expand or drop-down button is provided. The user cannot expand/contract expandable column and cannot drop-down a menu.
cesDropDown	The column title displays a drop-down button that the user can click to drop-down a menu. Clicking the drop-down button triggers an OnExpandClick event.
cesDropDownMenu	The column title displays a drop-down button that the user can click to drop-down a menu. Clicking the drop-down button the DropDownMenu is automatically dropped down.
cesFilterList	The column title displays a drop-down button that the user can click to drop-down a filter list. Clicking the drop-down button the filter DropDownList is automatically dropped down. {* ver. 6.0 *}
cesFilterForm	The column title displays a drop-down button that the user can click to drop-down a filter form. Clicking the drop-down button the filter DropDownForm is automatically dropped down. {* ver. 7.1 *}

See also: TXColumn.ButtonStyle, TXColumn.Expandable, TXColumn.Expanded, TXColumn.ExpandCols, TXCustomDBGrid.OnExpandClick, TXCustomDBGrid.OnColExpand, TXColumnTitle.DropDownMenu

TXColumn.ExpandWidth

Specifies the total width of all expanded columns.

```
property ExpandWidth: Integer;
```

Description
When column is Expandable, ExpandWidth specifies the total width of all Expanded columns.

See also: TXColumn.ExpandIndex, TXColumn.ExpandStyle, TXColumn.ExpandCols, TXColumn.Expanded, TXColumn.Expandable, TADTField

TXColumn.FieldName

Indicates the name of the field represented by the column.

```
property FieldName: string;
property FieldName: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
Setting FieldName changes the Field property so that it points to the dataset field with the same name. If the dataset does not have a field with the same name, Field is set to nil (Delphi), or NULL (C++).

See also: TXColumn.KeyField, TXColumn.LookupKeyField, TXColumn.LookupResultField, TXColumn.LookupListFields

TXColumn.Filtered

Specifies whether the column is filtered.

```
property Filtered: Boolean; {* ver. 6.0 *}
```

Description
Filtered returns True when FilterList is not empty or FilterText contains filter condition. Otherwise Filtered returs False.

See also: TXColumn.FilterList, TXColumn.FilterText

TXColumn.FilterData

Holds data from filter form for the column.

```
property FilterData: TStrings; {* ver. 7.1 *}
```

Description
The FilterData property holds data from filter form for the column.

See also: TXColumn.Filtered, TXColumn.FilterList

TXColumn.FilterList

Lists values that are filtered for the column.

```
property FilterList: TStrings; {* ver. 6.0 *}
```

Description
The FilterList property points to a TStrings object. If ExpandStyle is cbsFilterList or AutoFilter is active, these strings are checked in the drop-down filter list associated with the column's title.

See also: TXColumn.Filtered, TXColumn.FilterText, TXColumn.PickList

TXColumn.FilterText

Holds filter condition for the column.

```
property FilterText: string; {* ver. 6.0 *}
```

Description
The FilterText property holds the filter condition for the column.

See also: TXColumn.Filtered, TXColumn.FilterList

TXColumn.Grid

Indicates the grid that contains the column.

```
property Grid: TXCustomDBGrid;
```

Description
Use the Grid property to access the grid that contains the column.

See also: TXDBGrid

TXColumn.Hint

Specifies the text string that can appear when the user moves the mouse pointer over a data cell.

```
property Hint: string;
```

Description
Set Hint to a string that provides information about data cells. The hint text appears in a Help Hint window when the user pauses with the mouse over the cell if HintOptions includes hoShowDataHints.

See also: TXCustomDBGrid.HintOptions, TXCustomDBGrid.OnCellHint, TXCustomDBGrid.FillerHint, TXCustomDBGrid.IndicatorHint, TXColumn.EditorHint, TXColumnTitle.Hint, TXColumnTotal.Hint

TXColumn.ImageDraw

Determines how the image is displayed in the column.

```
type
  TImageDraw = (idAlignment, idOpposite, idCenter, idScaled, idStretch,
    idTileCell, idTileGrid, idTileCol, idTileRow);
property ImageDraw: TImageDraw;
```

Description
Set the ImageDraw property to determine method of displaying for ftGraphic fields, check boxes and images. These are the possible values of ImageDraw:

Value	Meaning
idAlignment	The image is displayed according to Alignment and VAlignment properties settings.
idOpposite	The image is displayed opposite to Alignment and VAlignment properties settings.
idCenter	The image is centered in the cell.
idScaled	The image is scaled to the cell boundaries.
idStretch	The image is stretched to the cell boundaries.
idTileCell	The image is tiled starting from boundaries of each cell.
idTileGrid	The image is tiled starting from boundaries of the grid.
idTileCol	The image is tiled starting from boundary of each column.
idTileRow	The image is tiled starting from boundary of each row.

See also: TXColumn.ImageOffsetX, TXColumn.ImageOffsetY, TXColumn.Transparent, TXColumn.Wallpaper, TXColumn.QuickDraw

TXColumn.ImageLevel

Specifies number of additional margins on the left side of ImageMargin.

```
property ImageLevel: Integer; {* ver. 6.8 *}
```

Description
Use ImageLevel to determine number of additional margins on the left side of ImageMargin when ImageMargin is greater than 0. When ImageMargin is 0 the ImageLevel does not matter. The total size of additional margins is calculated as ImageLevel * ImageMargin. You can change ImageLevel property when OnCalcImageIndex event is fired for the column.

See also: TXColumn.ImageMargin

TXColumn.ImageMargin

Specifies width of the left margin in pixels to display image inside.

```
property ImageMargin: Integer; {* ver. 6.5 *}
```

Description
Use ImageMargin to determine width of the left margin for the grid's cell. When ImageMargin is greater than 0 the image is drawn inside the margin area. When ImageMargin is 0 the image is drawn inside whole cell.

Only when ImageMargin is greater then 0 and Images are defined, the image is also drawn inside InplaceEditor when it edits text in the cell.

See also: TXColumn.ImageDraw, TXColumn.Transparent, TXColumn.Wallpaper

TXColumn.ImageOffsetX

Specifies horizontal indent in pixels for displayed images.

```
property ImageOffsetX: Integer;
```

Description
Use ImageOffsetX to determine how far image is indented from left or/and right boundary of the cell. Meaning of the ImageOffsetX property is strong connected with value of ImageDraw property.

See also: TXColumn.ImageDraw, TXColumn.ImageOffsetY, TXColumn.Transparent, TXColumn.Wallpaper

TXColumn.ImageOffsetY

Specifies vertical indent in pixels for displayed images.

```
property ImageOffsetY: Integer;
```

Description
Use ImageOffsetY to determine how far image is indented from top or/and bottom boundary of the cell. Meaning of the ImageOffsetY property is strong connected with value of ImageDraw property.

See also: TXColumn.ImageDraw, TXColumn.ImageOffsetX, TXColumn.Transparent, TXColumn.Wallpaper

TXColumn.Images

Determines which image list is associated with the column.

```
property Images: TImageList;
```

Description
Use Images to provide a customized list of bitmaps that can be displayed in the column. Use properties: ImageDraw, ImageMargin, ImageOffsetX, ImageOffsetY, QuickDraw, Transparent, Wallpaper to determine place and method how the image should be displayed in the column. Use OnCalcImageIndex event to change default image index.

Write an OnPaintColumnCell event handler to display image using other graphic format (*.jpg, *.gif, *.wfm, etc.) unsupported by TImageList.

See also: TXColumn.TransparentColor, TXCustomDBGrid.TitleImages, TXCustomDBGrid.IndicatorImages

TXColumn.KeyField

Indicates the key field in the dataset to match when doing a lookup.

```
property KeyField: string; {* ver. 5.0 *}
property KeyField: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
You can use LookupDataSet, KeyField, LookupKeyField, LookupResultField, LookupListFields to create a lookup list for a fkData field associated with the column. If ButtonStyle is cbsAuto and LookupColumn is defined, the LookupListFields values from LookupDataSet appear in the drop-down list associated with the column.

LookupKeyField is the field in the LookupDataSet which value must match the KeyField associated with column. The field specified in LookupKeyField must be of the same type as the KeyField associated with column, or the lookup list can't work.

Matching the value of the LookupKeyField in the LookupDataSet with the value of KeyField associated with the column determines a specific record in the LookupDataSet when the lookup list is dropped down. When the lookup list is closed, the value of the LookupResultField in the indicated record is assigned to the field associated with the column.

See also: TXColumn.LookupDataSet, TXColumn.LookupKeyField, TXColumn.LookupResultField, TXColumn.LookupListFields, TXColumn.FieldName

TXColumn.ListOptions

Specifies various pick list, lookup list and other drop-down editors properties of the XDBGrid's column.

```
type
  TXListOption = (loAutoComplete, loAutoDropDown, loAutoCloseEditor,
    loSelectNextValue, loAllowClearValue, loShowToday, loShowTodayCircle,
    loShowWeekNumbers, loAutoTodayClose, loBoldSundays, loShowImages,
    loShowLookupListTitles, loAllowAdjustOrder, loAllowChangeOrder,
    loAllowColumnResize, loAllowColumnMoving, loExpandFirstColumn,
    loFirstColumnExpanded, loMultiSelectList, loAutoApplyList,
    loTreeViewCellExpand, loIncrementalList); {* ver. 4.3 *} {* ver. 5.0 *} {*
    ver. 6.0 *} {* ver. 6.5 *} {* ver. 6.7 *} {* ver. 7.1 *} {* ver. 7.8 *}
  TXListOptions = set of TXListOption;
property ListOptions: TXListOptions;
```

Description
Set ListOptions to include various pick list and lookup list properties for the column of XDBGrid. These are the possible values of ListOptions:

Value	Meaning
loAutoComplete	Specifies whether the edit box automatically completes words that the user types by selecting the first item that begins with the currently typed string. When loAutoComplete is True and PickList/PickText/LookupDataSet/Field.Lookup is available for the column, the edit box responds to user keystrokes by searching PickList/PickText/LookupDataSet/Field.LookupDataSet for the first item that matches the string entered so far. If it finds an item that begins with the prefix typed so far, the edit box selects that item, "completing" the user's typing. If the user continues to type, selection may move to another, later, item, as the currently typed prefix no longer matches the initial loAutoComplete selection. If the user types a string that is not the prefix of a string in the PickList/PickText/LookupDataSet/Field.LookupDataSet, the string is taken as a unique entry and none of the items in the list are selected.
loAutoDropDown	The list is automatically dropped down, when the user starts edit the cell. When loAutoDropDown is not included, the user can press F2 key (in EditorMode) to

	simply drop down the list or other drop-down editor. This option is allowed for all drop-down editors.
loAutoCloseEditor	When the list is closed up by using Enter Key or either by mouse click in selected list item, the editor box is automatically closed (when dgAlwaysShowEditor option is not included in Options). When the list is closed up by F2 or Esc key the editor box is never closed. This option is allowed for all drop-down editors.
loSelectNextValue	The user can select next/prior value from the list by using Grey Plus/Minus key, when the list is'nt dropped down. When loSelectNextValue is included, next value can be selected by double click too. This option is allowed only for pick and lookup lists.
loAllowClearValue	When loAllowClearValue is included, the user can clear value into the cell by using Delete key. This option is especially useful to enter empty value to the lookup field or clear field's value, when poDropDownList option is included to the PickOptions. This option is allowed only for pick and lookup lists.
loShowToday	Specifies whether today's date is shown in drop-down calendar. See also: TMonthCalendar.ShowToday. This option is allowed only for drop-down calendar.
loShowTodayCircle	Specifies whether today's date is circled on drop-down calendar. See also: TMonthCalendar.ShowTodayCircle. This option is allowed only for drop-down calendar.
loShowWeekNumbers	Specifies whether week numbers are shown to the left of drop-down calendar. See also: TMonthCalendar.WeekNumbers. This option is allowed only for drop-down calendar.
loAutoTodayClose	Specifies whether the mouse click on today's date shown in drop-down calendar closes the drop-down list. This option is allowed only for drop-down calendar.
loBoldSundays	Specifies whether bold sundays are shown in drop-down calendar. To bold other days in drop-down calendar write OnCalcBoldDays event handler. This option is allowed only for drop-down calendar. `{* ver. 6.0 *}`
loShowImages	Specifies whether Images are shown on drop-down list and filter list. To show Images on the list you may need to write OnCalcImageImdex event handler.`{* ver. 6.5 *}`
loShowLookupListTitles	Specifies whether lookup list shows titles for the columns. You can specify caption for each list column in LookupListTitles property. This option (and the following) are useful only when dgGridStyleList option is included in OptionsExt.
loAllowAdjustOrder	Specifies whether lookup list allows to automatic adjust order when lookup list is dropped-down. This option is useful only when lookup DataSet class is registered to can automatically change order. See RegisterChangeOrder procedure.
loAllowChangeOrder	Specifies whether lookup list allows to change order by the user when lookup list is displayed. This option is useful only when lookup DataSet class is registered to can automatically change order. See RegisterChangeOrder procedure.
loAllowColumnResize	Specifies whether lookup list allows to resize columns. You can specify width for each list column in LookupListWidths property. When this option is included, the user can resize columns in the list.
loAllowColumnMoving	Specifies whether lookup list allows to move columns in the list. When this option is included, the user can move columns in the list.
loExpandFirstColumn	Specifies whether first column in the lookup list is Expandable. When this option is included, the first column in lookup list is expandable and the other

	columns are the child columns. See also: ExpandCols. **Note**. As first expandable column should be always defined lookup column that is corresponding width column's field.
loFirstColumnExpanded	Specifies whether first expandable column in lookup list is Expanded. When the column is expanded the child columns are visible on the list.
loMultiSelectList	Specifies whether lookup list or pick list allows to multi select values. Check MultiSelectList property to read more about. {* ver. 6.7 *}
loAutoApplyList	Specifies whether MultiSelectList allows to automatically apply changes in DataSet linked to the grid. Exclude loAutoApplyList to show OK and Cancel button in the multi select list. {* ver. 6.7 *}
loTreeViewCellExpand	Specifies whether the TreeView list allows to expand/collapse rows by clicking on expand box only or either on the whole expandable cell. {* ver. 7.1 *}
loIncrementalList	Specifies whether the Incremental Search function is active for data list or pick list and filter list in the column. See also dgIncrementalList option in Options property and foIncrementalList in FilterGrid.Options. {* ver. 7.8 *}

See also: TXColumn.PickList, TXColumn.PickOptions, TXColumn.PickText, TXColumn.ButtonStyle, TXCustomDBGrid.OnCalcBoldDays

TXColumn.LookupActive

Specifies whether the column has a lookup list.

```
property LookupActive: Boolean; {* ver. 6.0 *}
```

Description
Use LookupActive property to determine the column has defined lookup list.

LookupActive is a read-only property.

See also: TXColumn.LookupColumn, TXColumn.CalendarActive, TXColumn.PickDataActive

TXColumn.LookupColumn

Determines the kind of the lookup column.

```
type
  TXLookupColumn = (lcNotDefined, lcLookupField, lcLookupKeyField,
    lcLookupResultField, lcLookupValueField {* ver. 6.2 *});
property LookupColumn: TXLookupColumn; {* ver. 5.0 *}
```

Description
Use LookupColumn property to read the kind of lookup column. These are the possible values of LookupColumn:

Value	Meaning
lcNotDefined	The column is not a lookup column.
lcLookupField	The column shows the lookup field (Column.Field.FieldKind = fkLookup).
lcLookupKeyField	The column shows the key field for LookupDataSet (Column.Field is a foreign key).
lcLookupResultField	The column shows the result field from LookupDataSet when Column.Field is compatible with Column.LookupResultField.
lcLookupValueField	The column shows any result field from LookupDataSet when Column.Field is a foreign key. {* ver. 6.2 *}

Important notice!
When LookupColumn is lcLookupValueField the Column.Field on DataSet level is still a foreign key field and you should store ID value to the field. On TXDBGrid level the field is presented as LookupResultField including DisplayName, DefaultAlignment, DefaultWidth and DefaultCaption for column's Title. `{* ver. 6.2 *}`

When TXDBGrid is sorted by column with lcLookupValueField the sort order is consistent with the order of the key value (retrived from database) not of the result value (presented in the grid). For FilterList and editor's DropDownList the sort order is consistent with the order of the result value. Compare all differences for lcLookupField, lcLookupResultField and lcLookupValueField for three Company columns in XDBGridsLookups example. `{* ver. 6.2 *}`

See also: TXColumn.LookupDataSet, TXColumn.KeyField, TXColumn.LookupKeyField, TXColumn.LookupResultField, TXColumn.LookupListFields, TXColumn.FieldName

TXColumn.LookupDataSet

Identifies the dataset used to look up values for fkData field associated with the column.

```
property LookupDataSet: TDataSet;
```

Description
You can use LookupDataSet, KeyField, LookupKeyField, LookupResultField, LookupListFields to create a lookup list for a fkData field associated with the column. If ButtonStyle is cbsAuto and LookupColumn is defined, the LookupListFields values from LookupDataSet appear in the drop-down list associated with the column.

Use LookupDataSet to specify the dataset to use for looking up field values in a field (associated with the column) with a FieldKind of fkData. TXColumn.LookupDataSet for a FieldKind of fkData is similar to TField.LookupDataSet for a FieldKind of fkLookup.

When the lookup list is dropped down, the record in the LookupDataSet that contains the correct value is found by matching the LookupKeyField in the LookupDataSet with the current value of the KeyField associated with the column.

When the lookup list is dropped down and LookupDataSet is closed, it's automatically opened and then closed, when the lookup list is closed. It allows to use always fresh data in the lookup list. When the lookup list is dropped down and LookupDataSet is early opened, no open/close action is performed. In this case the developer must decide when LookupDataSet should be refreshed.

See also: TXColumn.KeyField, TXColumn.LookupKeyField, TXColumn.LookupResultField, TXColumn.LookupListFields, TXColumn.FieldName

TXColumn.LookupKeyField

Identifies the field in the lookup dataset to match when doing a lookup.

```
property LookupKeyField: string;
property LookupKeyField: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description

You can use LookupDataSet, KeyField, LookupKeyField, LookupResultField, LookupListFields to create a lookup list for a fkData field associated with the column. If ButtonStyle is cbsAuto and LookupColumn is defined, the LookupListFields values from LookupDataSet appear in the list associated with the column.

LookupKeyField is the field in the LookupDataSet which value must match the KeyField associated with column. The field specified in LookupKeyField must be of the same type as the KeyField associated with column, or the lookup list can't work.

Matching the value of the LookupKeyField in the LookupDataSet with the value of KeyField associated with the column determines a specific record in the LookupDataSet when the lookup list is dropped down. When the lookup list is closed, the value of the LookupResultField in the indicated record is assigned to the field associated with the column.

See also: TXColumn.LookupDataSet, TXColumn.KeyField, TXColumn.LookupResultField, TXColumn.LookupListFields, TXColumn.FieldName

TXColumn.LookupListFields

Identifies the field or fields whose values are displayed in the lookup list.

```
property LookupListFields: string;
property LookupListFields: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description

You can use LookupDataSet, KeyField, LookupKeyField, LookupResultField, LookupListFields to create a lookup list for a fkData field associated with the column. If ButtonStyle is cbsAuto and LookupColumn is defined, the LookupListFields values from LookupDataSet appear in the list associated with the column.

LookupListFields is the name of the field or fields in the LookupDataSet that are displayed in the drop-down list. LookupListFields can represent more than one field.

Before specifying LookupListFields, specify the LookupResultField property. If LookupListFields is not set, the lookup list displays LookupResultField field value by default. TXColumn.LookupListFields is similar to TDBLookupControl.ListField.

You can also use LookupListFields property when the field associated with the column is a fkLookup field. Then, you can specify more than one field (instead of Field.LookupResultField) displayed in the drop-down list. For a fkLookup field associated with the column the LookupListFields should be specified from Field.LookupDataSet.

See also: TXColumn.LookupDataSet, TXColumn.KeyField, TXColumn.LookupResultField, TXColumn.LookupKeyField, TXColumn.FieldName

TXColumn.LookupListTitles

Specifies the titles for lookup list columns.

```
property LookupListTitles: string; {* ver. 4.3 *}
```

Description
You can show titles for lookup list columns by including loShowLookupListTitles option in ListOptions. The LookupListTitles property specifies titles (separated by semicolor) for each column (field) specified in LookupListFields. When this property is empty the field name is showing as title for each lookup list column.

See also: TXColumn.LookupDataSet, TXColumn.LookupKeyField, TXColumn.LookupListFields, TXColumn.LookupListWidths

TXColumn.LookupListWidths

Specifies the widths for lookup list columns.

```
property LookupListWidths: string; {* ver. 4.3 *}
```

Description
You can show titles for lookup list columns by including loShowLookupListTitles option in ListOptions. The LookupListWidths property specifies widths (separated by semicolor) for each column (field) specified in LookupListFields. When this property is empty the default width is setting for each lookup list column.

See also: TXColumn.LookupDataSet, TXColumn.LookupKeyField, TXColumn.LookupListFields, TXColumn.LookupListTitles

TXColumn.LookupParentField

Indicates the parent field for tree view in the lookup dataset.

```
property LookupParentField: string; {* ver. 7.0 *}
property LookupParentField: WideString; // Delphi 2006, 2007 Win32, 2009,
2010, XE only
```

Description
LookupParentField contains key value of parent row for tree view on the lookup list. LookupParentField is the field in the LookupDataSet which value must match the LookupKeyField. The field specified in LookupParentField must be of the same type as LookupKeyField, or the tree view in the lookup list can't work.

See also: TXColumn.LookupDataSet, TXColumn.LookupKeyField, TXColumn.LookupTreeViewFields

TXColumn.LookupResultField

Indicates the result field in the lookup dataset.

```
property LookupResultField: string; {* ver. 5.0 *}
property LookupResultField: WideString; // Delphi 2006, 2007 Win32, 2009,
2010, XE only
```

Description
You can use LookupDataSet, KeyField, LookupKeyField, LookupResultField, LookupListFields to create a lookup list for a fkData field associated with the column. If ButtonStyle is cbsAuto and LookupColumn is defined, the LookupListFields values from LookupDataSet appear in the drop-down list associated with the column.

LookupResultField is the field in the LookupDataSet which value must match the field associated with column. The field specified in LookupResultField must be of the same type as the field associated with

column, or the lookup list can't work. These rules are not applied when LookupColumn is lcLookupValueField. {* ver. 5.0 *}

Matching the value of the LookupKeyField in the LookupDataSet with the value of KeyField associated with the column determines a specific record in the LookupDataSet when the lookup list is dropped down. When the lookup list is closed, the value of the LookupResultField in the indicated record is assigned to the field associated with the column.

See also: TXColumn.LookupDataSet, TXColumn.KeyField, TXColumn.LookupKeyField, TXColumn.LookupListFields, TXColumn.FieldName

TXColumn.LookupTreeViewField

Indicates the field for tree view in the lookup dataset.

```
property LookupTreeViewField: string; {* ver. 7.0 *}
property LookupTreeViewField: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
LookupTreeViewField indicates field in LookupDataSet for the tree view on lookup list. LookupTreeViewField is one of the field on list LookupListFields.

See also: TXColumn.LookupDataSet, TXColumn.LookupListFields, TXColumn.LookupParentField

TXColumn.MultiSelectList

Indicates whether the column has associated multi select list.

```
property MultiSelectList: Boolean; {* ver. 6.7 *}
```

Description
MultiSelectList property indicates wherher the column has associated multi select list. MultiSelectList returs True only when loMultiSelectList option is included in ListOptions and DataType of column's field is string or memo and the column has associeted PickList/PickText or LookupColumn is lcLookupKeyField. When MultiSelectList is True the associated list allows to multi select values. Values selected on the list are stored to the field separated by DelimiterLine or DelimiterChar. Count of selected values are limited by maximum size of the field.

MultiSelectList is a readonly property.

See also: TXColumn.ListOptions, TXColumn.MultiSelectText

TXColumn.MultiSelectText

Specifies how to display multi selected values in the column.

```
type
  TMultiSelectText = (mstDefault, mstEllipsis);
property MultiSelectText: TMultiSelectText; {* ver. 6.7 *}
```

Description
When MultiSelectList is associated with the column, MultiSelectText property specifies how to display multi selected values. These are the possible values of MultiSelectText property:

Value	Meaning
mstDefault	Multi selected values are displayed exactly as they are stored in the field (separated by DelimiterLine or DelimiterChar).
mstEllipsis	Only first selected value is displayed and it is finished by ellipsis when more then one value was selected on the list.

See also: TXColumn.MultiSelectList, TXColumn.ListOptions

TXColumn.ParentColumn

Refers to the TXColumn object that owns the column.

```
property ParentColumn: TXColumn;
```

Description
If the column represents a child column, ParentColumn refers to the column that parents the child column.

See also: TXDBGrid

TXColumn.PickData

Determines a pick list currently used to show values for the column.

```
property PickData: TStrings; {* ver. 6.0 *}
```

Description
The PickData property points to PickList or either PickText or either formatted PickText list defined for the column depending on PickOptions and PickTextFormat properties. The PickData list always contains values currently visible in the grid.

PickData is a read-only property.

See also: TXColumn.PickDataList, TXColumn.PickList, TXColumn.PickText, TXColumn.PickTextFormat, TXColumn.PickOptions

TXColumn.PickDataActive

Specifies whether the column has a pick data list.

```
property PickDataActive: Boolean; {* ver. 6.0 *}
```

Description
Use PickDataActive property to determine the column has defined a pick data list.

PickDataActive is a read-only property.

See also: TXColumn.PickDataList, TXColumn.LookupActive, TXColumn.CalendarActive

TXColumn.PickDataList

Determines a pick list visibled as drop-down list for the column.

```
property PickDataList: TStrings; {* ver. 6.0 *}
```

Description
The PickDataList property points to PickList or either PickText or either formatted PickText list defined for the column depending on PickOptions and PickTextFormat properties. The PickDataList always contains values visibled on drop-down list.

PickDataList is a read-only property.

See also: TXColumn.PickData, TXColumn.PickList, TXColumn.PickText, TXColumn.PickTextFormat, TXColumn.PickOptions

TXColumn.PickIndex

Specifies the index of the current value in the pick data list.

```
property PickIndex: Integer; {* ver. 6.4 *}
```

Description
When PickDataActive property is True, you can use PickIndex to read index of the current value in the pick data list.

PickIndex is a read-only property.

See also: TXColumn.PickData, TXColumn.PickDataActive

TXColumn.PickList

Lists values that the user can select for the column.

```
property PickList: TStrings;
```

Description
The PickList property points to a TStrings object. If ButtonStyle is cbsAuto, these strings appear in the drop-down list associated with the column.

See also: TXColumn.LoadPickList, TXColumn.PickText, TXColumn.ListOptions, TXColumn.PickOptions

TXColumn.PickOptions

Specifies various PickList and PickText properties of the XDBGrid's column.

```
type
  TXPickOption = (poDropDownList, poListItemsOnly, poSelectPickText,
    poStoreItemIndex, poAcceptPickList, poAcceptItemIndex, poAutoLoadList,
    poAppendToList, poSortedByText, poFormatPickText, poFormatListOnly); {*
    ver. 4.3 *}
  TXPickOptions = set of TXPickOption;
property PickOptions: TXPickOptions;
```

Description
Set PickOptions to include various PickList and PickText properties for the column of XDBGrid. These are the possible values of PickOptions:

Value	Meaning
poDropDownList	Creates a drop-down list with no edit box. The user cannot enter text manually. See also ListOptions property.
poListItemsOnly	The user can enter text manually, but only values from the list are accepted. You can include this option instead of poDropDownList option.
poSelectPickText	When PickText list is not empty, values from PickText are displayed instead corresponding items values from PickList that are stored in dataset. This option can't be included with poAutoLoadList option.
poStoreItemIndex	When poSelectPickText option is included, the PickText.ItemIndex is stored to dataset instead of corresponding item value from PickList. The value stored in dataset is right justified for TStringField to determine order suitable to items order in PickText list.
poAcceptPickList	When poSelectPickText option is included, the user can also enter values from PickList in edit box.
poAcceptPickIndex	When poSelectPickText option is included, the user can also enter ItemIndex values in edit box. The list items are always numbered starting from 0.
poAutoLoadList	When poAutoLoadList option is included, the PickList items are updated on the basis current unique values for this column. The PickList items are automatically updated by using UpdateListItems method when dgAutoUpdateListItems option is included in OptionsExt. See also LoadPickList method.
poAppendToList	When poAppendToList option is included, the UpdateListItems, LoadPickList, LoadPickText methods appends new values to the PickList/PickText. When this option is excluded (default), the PickList/PickText is cleared before it's loaded. You should include this option to prevent PickList/PickText items defined in design-time.
poSortedByText	When poSortedByText option is included, the LoadPickText method (with parameter Sorted = True) loads values pair to the PickList/PickText objects in alphabetical order according to PickText items order, otherwise - according to PickList items order.
poFormatPickText	When poFormatPickText option is included, the FormatPickText function is performed before PickText items are displayed. You can define PickTextFormat string for FormatPickText function. If the string specified by the PickTextFormat property is empty, the DefPickTextFormat variable will be used.
poFormatListOnly	When poFormatListOnly option is included, the FormatPickText function is

performed only for dropdown list values. The value in edit box is not formatted.

See also: TXColumn.PickList, TXColumn.PickText, TXColumn.ListOptions

TXColumn.PickText

Determines text for corresponding PickList's values that the user can select for the column.

```
property PickText: TStrings;
```

Description
The PickText property points to a TStrings object. If poSelectPickText option is included in PickOptions and PickText is not empty, values from PickText are displayed instead corresponding values from PickList (stored in dataset). If ButtonStyle is cbsAuto, these strings appear in the drop-down list associated with the column.

See also: TXColumn.LoadPickText, TXColumn.PickList, TXColumn.ListOptions, TXColumn.PickOptions

TXColumn.PickTextFormat

Specifies format string for PickText list formatting.

```
property PickTextFormat: string; {* ver. 4.3 *}
```

Description
Use PickTextFormat property to determine format string for PickText items. When poFormatPickText option is included in PickOptions the PickText items are formatted by using FormatPickText function.

If the string specified by the PickTextFormat property is empty, the PickText item value is formatted according to DefPickTextFormat variable.

When PickTextFormat is not empty it must contain format string (any text) with one, two or three argument specifiers. The each argument specifier must be appropriate to expected type according to SysUtils.Format function for strings and numeric values in order: **string**, **string**, **Integer**. The result of FormatPickText function is performed as:

```
Result := Format(PickTextFormat, [PickTextItem, PickListItem, ListIndex];
```

Use property editor to select one of most popular format for PickTextFormat. These are the most popular formats:

Format	Result
%0:s - %1:s	Kansas - KA
%0:s (%1:s)	Kansas (KA)
%0:s (%2:d)	Kansas (18)
%1:s - %0:s	KA - Kansas
%1:s (%0:s)	KA (Kansas)
%2:2d. %0:s	18. Kansas
%2:2d. %0:s (%1:s)	18. Kansas (KA)
%2:2d. %1:s (%0:s)	18. KA (Kansas)

See also: TXColumn.PickText, TXColumn.FormatPickText, TXColumn.PickOptions, DefPickTextFormat

TXColumn.QuickDraw

Specifies whether the image is displayed using a palette.

```
property QuickDraw: Boolean;
```

Description
Set QuickDraw to specify whether a customized palette should be used when displaying field values. If False, a palette is used, to provide the best possible image quality at the expense of additional processing time. If True, no special palette is used, which is faster, but results in poorer picture quality, especially with 256-color images on a 256-color video driver.

See also: TXColumn.ImageDraw

TXColumn.Report

Indicates the TXColumnReport that determines additional column's properties for reporting.

```
property Report: TXColumnReport;
```

Description
The Report property points to a TXColumnReport object that determines additional attributes of the column needed for XQRGrid component.

See also: TXColumnReport

TXColumn.ShowBlob

Determines whether graphics and memos can be displayed in the column.

```
property ShowBlob: Boolean;
```

Description
If ShowBlob is set to True, ftGraphic, ftMemo and ftFmtMemo fields can be displayed in the column.

See also: TXColumn.ImageDraw, TXColumn.ShowEdit

TXColumn.ShowEdit

Determines whether data in the column may be displayed in edit mode.

```
property ShowEdit: Boolean;
```

Description
If ShowEdit is set to False, data in the column cannot be displayed by the user in edit mode. This property is especially useful for display ftGraphic fields in the grid.

See also: TXColumn.ImageDraw, TXColumn.ShowBlob, TXColumn.EditMemo

TXColumn.SuppressRTL

Suppress the bi-directional mode for the column.

```
property SuppressRTL: Boolean;
```

Description
When BiDiMode is set to bdRightToLeft this property allow to suppress right to left reading for the column.

See also: TControl.BiDiMode, TXColumn.UseRightToLeftAlignment

TXColumn.Title

Indicates the TXColumnTitle that represents the column's title.

```
property Title: TXColumnTitle;
```

Description
The Title property points to a TXColumnTitle object that determines attributes of the column's title. If FieldName is set, the value of FieldName becomes the default column title (TXColumnTitle.Caption).

The title (TXColumnTitle.Caption) appears at runtime only if the dgTitles flag is set in the data grid's Options property.

See also: TXColumnTitle

TXColumn.ToolTips

Specifies whether the cells in the column have tooltips.

```
property ToolTips: Boolean;
```

Description
Set ToolTips to True to specify that cells in the column have tooltips (Help Hints). See also HintOptions.

See also: TXColumnTotal.ToolTips, TXColumn.ToolTipsWidth, TXCustomDBGrid.HintOptions

TXColumn.ToolTipsWidth

Specifies how the width of tooltips window is calculated.

```
type
  TToolTipsWidth = (ttAutoWidth, ttFixedWidth);
property ToolTipsWidth: TToolTipsWidth;
```

Description
Set the ToolTipsWidth property to determine width of tooltips window. These are the possible values of ToolTipsWidth:

Value	Meaning
ttAutoWidth	The width of tooltips window is calculated according to width of field's text.
ttFixedWidth	The width of tooltips window is calculated according to width of column.

Notice. When the column represents memo field (ftMemo or ftFmtMemo) then ToolTipsWidth property is ignored. In that case ttFixedWidth is always used.

See also: TXColumn.ToolTips, TXCustomDBGrid.HintOptions

TXColumn.TotalFields

Indicates the TXColumnTotalFields object that represents the total cells with fields values retrieved from Totals.DataSource.

```
property TotalFields: TXColumnTotalFields;
```

Description
The TotalFields property points to a TXColumnTotalFields object that determines attributes of the column's total cells with fields values retrieved from Totals.DataSource. The TotalFields cells are shown beetwen TotalHeader and TotalFooter cells. Count of TotalFields rows is determined by RecordCount property for Totals.DataSource.DataSet or by Totals.DataMaxRow property.

The TotalFields cells appear only if the dgTotalFields flag is set in the data grid's OptionsEx property.

See also: TXColumn.TotalFooter, TXColumn.TotalHeader, TXColumn.TotalValues, TXColumn.TotalRows

TXColumn.TotalFooter

Indicates the TXColumnTotalFooter object that determines the total cell with calculated value for last total row in the column.

```
property TotalFooter: TXColumnTotalFooter;
```

Description
The TotalFooter property points to a TXColumnTotalFooter object that determines attributes of the column's total cell with calculated value for last total row in the column.

The TotalFooter's cells appear only if the dgTotalFooter flag is set in the data grid's OptionsEx property.

See also: TXColumn.TotalFields, TXColumn.TotalHeader, TXColumn.TotalValues, TXColumn.TotalValueRows

TXColumn.TotalHeader

Indicates the TXColumnTotalHeader object that determines the total cell with calculated value for first total row in the column.

```
property TotalHeader: TXColumnTotalHeader;
```

Description
The TotalHeader property points to a TXColumnTotalHeader object that determines attributes of the column's total cell with calculated value for first total row in the column.

The TotalHeader's cells appear only if the dgTotalHeader flag is set in the data grid's OptionsEx property.

See also: TXColumn.TotalFooter, TXColumn.TotalFields, TXColumn.TotalValues, TXColumn.TotalValueRows

TXColumn.TotalRows

Specifies the TXColumnTotal object depend on current total row number.

```
property TotalRows[Index: Integer]: TXColumnTotal;
```

Description
Use TotalRows to retrieve TXColumnTotal object for specified total row number. Index determines current total row number starting from 0. Only visibled total rows are numbered. The result can be nil if Index is outside visibled rows range or when TotalValues for this column has undefined (empty) total cell for specified total row. See also TotalValueRows property.

TotalRows is a read-only property.

See also: TXColumn.TotalValueRows, TXColumn.TotalFields

TXColumn.TotalValueRows

Specifies the TXColumnTotalValue object depend on current total row number.

```
property TotalValueRows[Index: Integer]: TXColumnTotalValue;
```

Description
Use TotalValueRows to retrieve TXColumnTotalValue object for specified total row number. Index determines current total row number starting from 0. Only visibled total rows are numbered. The result can be nil if Index is outside visibled rows range or when TotalValues for this column has undefined (empty) total cell for specified total row or when Index points to TotalFields cell (not derived from TXColumnTotalValue). See also TotalRows property.

TotalValueRows is a read-only property.

See also: TXColumn.TotalRows, TXColumn.TotalValues, TXColumn.TotalFooter, TXColumn.TotalHeader

TXColumn.TotalValues

Indicates the TXColumnTotalFooter object that represents a collection of total cells with calculated values for the column.

```
property TotalValues: TXColumnTotalValues;
```

Description
The TotalValues property points to a TXColumnTotalValues collection of TXColumnTotalValue objects that determine attributes of the column's total cells with calculated values in the column. The TotalValues cells are shown beetwen TotalHeader and TotalFooter cells. Count of TotalValues rows is determined by maximum value of TotalValues.Count property for all columns in the grid.

The TotalValues cells appear only if the dgTotalValues flag is set in the data grid's OptionsEx property.

See also: TXColumn.TotalFooter, TXColumn.TotalHeader, TXColumn.TotalFields, TXColumn.TotalValueRows

TXColumn.Transparent

Specifies whether the background of the image is displayed.

```
property Transparent: Boolean;
```

Description
Use Transparent to specify that the graphic be drawn transparently. Some descendants of TGraphic such as TIcon and TMetafile are always transparent, so setting the property for those objects does not change their behavior. However, the TBitmap graphic's drawing is affected by this property.

See also: TXColumn.ImageOffsetX, TXColumn.ImageOffsetY, TXColumn.ImageDraw, TXColumn.Wallpaper

TXColumn.TransparentColor

Determines which color of the bitmap is to be transparent when the bitmap is drawn.

```
property TransparentColor: TColor;
```

Description
Use TransparentColor to determine how to draw the bitmap transparently. When the TransparentColor property is set to clNone (default), the bottom leftmost pixel shown onscreen is using as transparent color.

The TransparentColor property is allowed only for Images property. Value of this property should be according to transparent color using when images were added to ImageList. All images should then have common transparent color.

See also: TXColumn.Images

TXColumn.TrueWidth

Specifies the column width for invisible columns.

```
property TrueWidth: Integer;
```

Description
Use TrueWidth to get column width when column's Visible property is False (Width = -1).

TrueWidth is a read-only property.

See also: TColumn.Width, TXColumn.WidthBase, TXColumn.WidthMax, TXColumn.WidthMin

TXColumn.VAlignment

Specifies how text is vertical aligned within the cell.

```
property VAlignment: TVAlignment;
```

Description
Use VAlignment to specify whether the text is top-justified, bottom-justified, or centered.

See also: TColumn.Alignment, TXColumnTitle.VAlignment, TXColumnTotal.VAlignment

TXColumn.Visibility

Specifies whether the column is visible in the grid.

```
property Visibility: Integer;
```

Description
Use Visibility property to read internal state of column visibility when ParentColumn is contracted and child columns are invisible. The Visibility property returns a future state of the column visibility when ParentColumn will be Expanded. To show/hide a column in the grid use Visible property. The Showing property reflects the actual displayability of the column based on additional factors.

Visibility is a read-only property.

See also: TColumn.Visible, TColumn.Showing, TXColumn.ParentColumn, TXColumn.Expandable, TXColumn.Expanded, TXColumn.ExpandCols

TXColumn.Wallpaper

Specifies whether the field's text is displayed on the image.

```
property Wallpaper: Boolean;
```

Description
Use Wallpaper to specify that the field's text be drawn transparently on the image. This property has affect only when the image is displayed in the cell.

See also: TXColumn.ImageOffsetX, TXColumn.ImageOffsetY, TXColumn.ImageDraw, TXColumn.Transparent

TXColumn.WidthBase

Specifies the base column width.

```
property WidthBase: Integer;
```

Description
Use WidthBase to get or set the base column width when StretchMode is True. When StretchMode is False than value of WidthBase is equal Width.

See also: TColumn.Width, TXColumn.TrueWidth, TXColumn.WidthMax, TXColumn.WidthMin

TXColumn.WidthMax

Specifies the maximum column width.

```
property WidthMax: Integer;
```

Description
Use WidthMax to get or set the maximum column width.

See also: TColumn.Width, TXColumn.TrueWidth, TXColumn.WidthBase, TXColumn.WidthMin

TXColumn.WidthMin

Specifies the minimum column width.

```
property WidthMin: Integer;
```

Description
Use WidthMin to get or set the minimum column width.

See also: TColumn.Width, TXColumn.TrueWidth, TXColumn.WidthBase, TXColumn.WidthMax

TXColumn.WordWrap

Specifies whether the data wraps when it is too long for the width of the column.

```
property WordWrap: Boolean;
```

Description
Set WordWrap to True to allow the data to display multiple line of text. When WordWrap is True, text that is too wide for the column wraps at the right margin and continues in additional lines.

Set WordWrap to False to limit the data to a single line. When WordWrap is False, text that is too wide for the column appears truncated. See also Ellipsis property.

See also: TXColumn.Ellipsis, TXColumnTitle.WordWrap, TXColumnTotal.WordWrap

TXColumn.DesignedWidth

Returns the designed width for the column.

```
function DesignedWidth: Integer; {* ver. 8.0 *}
```

Description
The value of DesignedWidth returns designed width of the column in Delphi IDE or either DefaultWidth.

See also: TColumn.DefaultWidth, TXColumn.OptimalWidth

TXColumn.FormatPickText

Formats a PickText item value.

```
procedure FormatPickText(ListIndex: Integer): string; {* ver. 4.3 *}
```

Description
FormatPickText formats the PickText item value given by ListIndex using the format given by PickTextFormat property.

If the string specified by the PickTextFormat property is empty, the PickText item value is formatted according to DefPickTextFormat variable.

When PickTextFormat is not empty it must contain format string (any text) with one, two or three argument specifiers. The each argument specifier must be appropriate to expected type according to SysUtils.Format function for strings and numeric values in order: **string**, **string**, **Integer**. The result of FormatPickText function is performed as:

```
  Result := Format(PickTextFormat, [PickTextItem, PickListItem, ListIndex];
```

See also: TXColumn.PickText, TXColumn.PickTextFormat, DefPickTextFormat

TXColumn.IsCheckBoxValue

Indicates whether the field's value is according to checkbox state.

```
function IsCheckBoxValue(Checked: Boolean): Boolean;
```

Description
IsCheckBoxValue returns True if the field's value is according to Checked parameter. See CheckBoxValues property.

See also: TXColumn.CheckBox, TXColumn.CheckBoxToggle, TXColumn.CheckBoxValues, TXColumn.ToggleCheckBoxValue

TXColumn.IsExpandableBox

Indicates whether the expand box is expandable for the row.

```
function IsExpandableBox: Boolean; {* ver. 6.8 *}
```

Description
IsExpandableBox returns True when the current row is expandable.

See also: TXColumn.ExpandBox, TXColumn.IsExpandedBox

TXColumn.IsExpandedBox

Indicates whether the expand box is expanded for the row.

```
function IsExpandedBox: Boolean; {* ver. 6.6 *}
```

Description
IsExpandedBox returns True if the expand form is open for the current row.

See also: TXColumn.ExpandBox, TXColumn.ExpandBoxToggle, TXColumn.ToggleExpandBox

TXColumn.LoadPickList

Loads unique items from dataset to the PickList.

```
procedure LoadPickList(PickListField: TField = nil; Sorted: Boolean = True);
{* ver. 4.3 *}
```

Description
Use LoadPickList method to load all unique values from PickListField's dataset to the PickList object. When PickListField parameter is nil, the Field linked to this column will be use to load data. You can specify PickListField from any dataset (not only from dataset linked to the grid). When PickListField's dataset is not active, the dataset will be open and close after that.

When Sorted parameter is True, the PickList items are sorted in alphabetical order. When Sorted is False, the PickList items are loaded according to current order in dataset.

Note. When poAppendToList option is included in PickOptions, the LoadPickList appends only new values to the PickList.

See also: TXColumn.PickList, TXColumn.LoadPickText

TXColumn.LoadPickText

Loads unique items pair from dataset to the PickList and PickText.

```
procedure LoadPickText(PickListField, PickTextField: TField; Sorted: Boolean =
True); {* ver. 4.3 *}
```

Description
Use LoadPickText method to load all unique values pair from PickListField/PickTextField's dataset to the PickList and PickText object. When PickListField or PickTextField parameter is nil, the PickList and PickText properties are cleared. You can specify PickListField/PickTextField from any dataset (not only from dataset linked to the grid), but both fields must come from the same dataset. When dataset is not active, the dataset will be open and close after that.

When Sorted parameter is True, the PickList/PickText items are sorted in alphabetical order. When poSortedByText option is included in PickOptionsthe alphabetical order depend on PickText items, otherwise the alphabetical order depend on PickList items. When Sorted is False, the PickList/PickText items are loaded according to current order in dataset.

Note. When poAppendToList option is included in PickOptions, the LoadPickText appends only new values pair to the PickList/PickText.

See also: TXColumn.PickText, TXColumn.LoadPickList

TXColumn.MoveTo

Moves the column in the Columns collection.

```
procedure MoveTo(ToIndex: Integer);
```

Description
Use MoveTo to move the column to the ToIndex position in Columns collection. You can also set the value of Index property to move the column, but MoveTo method moves also appropriate column in MasterGrid when MasterGrid property is assigned.

See also: TCollectionItem.Index, TXColumn.Synchronize, TXCustomDBGrid.MasterGrid

TXColumn.OptimalWidth

Returns the optimal width for the column.

```
function OptimalWidth: Integer;
```

Description
The value of OptimalWidth is determined by current field values visibled in the grid. When roOptimalWidth option is included in ResizeOption this function is called to calculate current optimal width of the column.

The result of OptimalWidth function also takes into account the width of the column title when dgTitleWidthOff option is excluded from Options property. {* ver. 7.5 *}

See also: TColumn.DefaultWidth, TXColumn.ExpandWidth, TXCustomDBGrid.ResizeOptions

TXColumn.SwitchListOptions

Include or exclude ListOptions depending on State.

```
procedure SwitchListOptions(Part: TXListOptions; State: Boolean); {* ver. 6.0 *}
```

Description
Use SwitchListOptions to include or exclude the Part of ListOptions property. When State is True the Part options are included, otherwise excluded.

See also: TXColumn.SwitchPickOptions

TXColumn.SwitchPickOptions

Include or exclude PickOptions depending on State.

```
procedure SwitchPickOptions(Part: TXPickOptions; State: Boolean); {* ver. 6.0 *}
```

Description
Use SwitchPickOptions to include or exclude the Part of PickOptions property. When State is True the Part options are included, otherwise excluded.

See also: TXColumn.SwitchListOptions

TXColumn.Synchronize

Fits width of the column to width of appropriate column in MasterGrid.

```
procedure Synchronize;
```

Description
Use Synchronize when MasterGrid is assigned and you made any special changes for this column you need call this method.

See also: TXCustomDBGrid.Synchronize

TXColumn.ToggleCheckBoxValue

Changes the state of the check box.

```
procedure ToggleCheckBoxValue;
```

Description
Use the ToggleCheckBoxValue method to select or deselect the check box programmatically.

See also: TXColumn.CheckBox, TXColumn.CheckBoxToggle, TXColumn.CheckBoxValues, TXColumn.IsCheckBoxValue

TXColumn.ToggleExpandBox

Expand form for the current row.

```
procedure ToggleExpandBox; {* ver. 6.6 *}
```

Description
Call ToggleExpandBox method to expand form for the current row.

See also: TXColumn.ExpandBox, TXColumn.ExpandBoxToggle, TXColumn.IsExpandedBox

TXColumn.UseRightToLeftAlignment

Specifies whether the column's alignment is in a right-to-left mode.

```
function UseRightToLeftAlignment: Boolean; {* ver. 6.0 *}
```

Description
Call UseRightToLeftAlignment to determine whether the column's alignment is in a right-to-left mode.

UseRightToLeftAlignment returns True for middle east locales if the BiDiMode property is bdRightToLeft and SuppressRTL property is False. Otherwise, it returns False.

See also: TControl.BiDiMode, TXColumn.SuppressRTL

TXColumnTitle

TXColumnTitle represents the title of a data-grid column (TXColumn).

Unit
XDBGrids

Description
TXDBGrid uses a TXDBGridColumns to maintain a collection of TXColumn objects. Each TXColumn has an associated TXColumnTitle that holds information about its title. The TXColumnTitle instance is stored in the column's Title property.

See also: TColumnTitle, TXColumn, TXColumn.Title, TXDBGridColumns, TXDBGrid

TXColumnTitle.AutoFilter

Determines whether the column has auto-filter list.

```
property AutoFilter: Boolean; {* ver. 6.3 *}
```

Description
Set AutoFilter to True to show auto-filter button in the column title that allows to drop down filter list for the column.

Set AutoFilter to False to prevent showing filter list for the column.

See also: TXDBGridFilter.AutoFilter

TXColumnTitle.AutoFilters

Specifies the available filter styles for the column.

```
property AutoFilters: TFilterStyles; {* ver. 7.1 *}
```

Description
Set AutoFilters to include available filter styles for the column. The filter style is available for the column only if it is included in this property as well as in the FilterGrid.AutoFilters property.

See also: TXDBGridFilter.AutoFilters, TXColumn.AutoFilter

TXColumnTitle.AutoToggle

Determines whether the sorting marker is toggled automatically when the user clicks the title of column.

```
property AutoToggle: Boolean;
```

Description
Set AutoToggle to True to cause the sorting marker is toggled automatically when the user clicks the title of column, if Options includes dgMarkerAutoToggle.

Set AutoToggle to False if the sorting marker for this column should be controlled programmatically. To toggle marker when AutoToggle is False, use the ToggleMarker method.

See also: TXColumnTitle.ToggleMarker, TXColumnTitle.Marker, TXCustomDBGrid.Options, TCustomDBGrid.OnTitleClick

TXColumnTitle.Button

Determines whether the title of column is animated like button when the user clicks on it.

```
property Button: Boolean;
```

Description
Set Button to True to cause the title of column is animated like button when the user clicks on the title of column, if Options includes dgTitleButtons.

Set Button to False if the title of this column should'nt be animated and OnTitleClick event should't be fired when the user clicks on this title of column, if Options includes dgTitleButtons.

If Options does'nt include dgTitleButtons this property has no effect.

See also: TXCustomDBGrid.Options, TCustomDBGrid.OnTitleClick

TXColumnTitle.Column

Specifies the TXColumn object that is associated with the column title.

```
property Column: TXColumn;
```

Description
Use Column to determine the column associated with the column title.

See also: TXColumn.Title

TXColumnTitle.DropDownMenu

Identifies the drop-down menu associated with the title of column.

```
property DropDownMenu: TPopupMenu;
```

Description
Assign a value to DropDownMenu to make a drop-down menu appear when the user clicks the expand button on the title of column. If the ExpandStyle property is cesAuto when DropDownMenu is assigned, the ExpandStyle property is automatically changed to cesDropDownMenu and the expand button appears in the title of column. If the ExpandStyle is cesDropDownMenu when DropDownMenu is removed, the ExpandStyle property is automatically changed to cesAuto. See also: DropDownPoint.

See also: TXColumnTitle.PopupMenu, TXColumn.DropDownMenu, TXColumn.ExpandStyle, TXCustomDBGrid.FillerPopupMenu, TXCustomDBGrid.IndicatorPopupMenu

TXColumnTitle.Ellipsis

Specifies whether the Caption is completed by ... when it is too long for the width of the column.

```
property Ellipsis: Boolean;
```

Description
Set Ellipsis to True to allow the Caption completed by triple dot. When Ellipsis is True, text that is too wide for the column is completed by triple dot.

When Ellipsis is False, text that is too wide for the column appears truncated. See also WordWrap property.

See also: TXColumn.Ellipsis, TXColumnTotal.Ellipsis, TXColumnTitle.WordWrap

TXColumnTitle.Header

Specifies the text that appears at the top of the title of column.

```
property Header: string;
```

Description
The Header property contains a text string that appears at the top of the titles. If the dgTitleHeaders flag is set in the Options property, header's rows are visible in the grid. Headers for adjacent columns are merged while rows contain identical text.

Use '|' character as separator to enter header's text for few rows in run-time. Use Header Editor to enter header's text for few rows in design-time.

See also: TColumnTitle.Caption, TXColumnTitle.HeaderRows, TXColumnTitle.HeaderHintRows

TXColumnTitle.HeaderHint

Specifies the text string that can appear when the user moves the mouse pointer over a header of titles.

```
property HeaderHint: string;
```

Description
Set HeaderHint to a string that provides more information about the meaning of the header than the Header property. The hint text appears in a Help Hint window when the user pauses with the mouse over the header if HintOptions includes hoShowTitleHints. In design time you may provide separate hint for all rows of header by using Header editor. In run-time you should follow a hint for each row by a vertical bar (|).

See also: TXColumnTitle.HeaderHintRows, TXColumnTitle.Hint, TXColumnTitle.HeaderRowCount, TXCustomDBGrid.OnCellHint

TXColumnTitle.HeaderHintRows

Contains the strings that appear in hint window when the user moves the mouse pointer over the rows of header.

```
property HeaderHintRows[Index: Integer]: string;
```

Description
Use HeaderHintRows to read hint text for single row of header. The HeaderHintRows property always return part of HeaderHint property. The rows are indexed from 0 to HeaderRowCount-1.

HeaderHintRows is read-only property.

See also: TXColumnTitle.HeaderHint, TXColumnTitle.HeaderRows, TXColumnTitle.HeaderRowCount

TXColumnTitle.HeaderRowCount

Specifies number of rows for the header.

```
property HeaderRowCount: Integer;
```

Description
Use HeaderRowCount to read number of rows for the header. The HeaderRowCount is calculated upon Header property value.

HeaderRowCount is read-only property.

See also: TXColumnTitle.Header, TXColumnTitle.HeaderRows, TXColumnTitle.HeaderHintRows

TXColumnTitle.HeaderRows

Contains the strings that appear in the rows of header.

```
property HeaderRows[Index: Integer]: string;
```

Description
Use HeaderRows to read caption for single row of header. The HeaderRows property always return part of Header property. The rows are indexed from 0 to HeaderRowCount-1.

HeaderRows is read-only property.

See also: TXColumnTitle.HeaderRowCount, TXColumnTitle.HeaderRows, TXColumnTitle.HeaderHintRows

TXColumnTitle.Hint

Specifies the text string that can appear when the user moves the mouse pointer over a title of column.

```
property Hint: string;
```

Description
Set Hint to a string that provides more information about the meaning of the column than the Caption. The hint text appears in a Help Hint window when the user pauses with the mouse over the title of column if HintOptions includes hoShowTitleHints.

See also: TColumnTitle.Caption, TXColumnTitle.HeaderHint, TXCustomDBGrid.FillerHint, TXCustomDBGrid.IndicatorHint, TXColumn.EditorHint, TXColumn.Hint, TXColumnTotal.Hint, TXCustomDBGrid.OnCellHint

TXColumnTitle.ImageIndex

Indicates which image maintained by the TitleImages appears to the left side of title's caption.

```
property ImageIndex: Integer;
```

Description
Set ImageIndex to designate an image that should appear to the left side of title's caption. The ImageIndex specifies a zero-offset index into the TitleImages property.

See also: TXCustomDBGrid.TitleImages, TXColumnTotal.ImageIndex

TXColumnTitle.Marker

Specifies sorting marker kind.

```
type
  TMarker = (tmNone, tmAscend, tmDescend);
property Marker: TMarker;
```

Description
Set the Marker property to determine kind of sorting marker. These are the possible values of Marker:

Value	Meaning
tmNone	None sorting marker is displayed in the title of column
tmAscend	Sorting marker indicate ascending order for the column
tmDescend	Sorting marker indicate descending order for the column

Options dgMarkerAutoSwitch, dgMarkerAutoToggle and dgMarkerAscendOnly included in Options are allowed when the Marker property is changed.

See also: TXColumnTitle.AutoToggle, TXColumnTitle.ToggleMarker, TXColumnTitle.MarkerIndex, TColumnTitle.OrderIndex

TXColumnTitle.MarkerIndex

Indicates which image maintained by the Markers appears to the right side of title's caption.

```
property MarkerIndex: Integer;
```

Description
Set MarkerIndex to designate an image that should appear to the right side of title's caption. The MarkerIndex specifies a zero-offset index into the Markers property. Value of MarkerIndex property can be changed automatically by setting value for Marker or OrderIndex property.

Options dgMarkerAutoSwitch, dgMarkerAutoToggle and dgMarkerAscendOnly included in Options are allowed when the MarkerIndex property is changed.

See also: TXColumnTitle.Marker, TXColumnTitle.OrderIndex

TXColumnTitle.OrderIndex

Identifies column's position in the OrderFields property.

```
property OrderIndex: Integer;
```

Description
Set OrderIndex to add or remove column to/from OrderFields property or to change it's position in the list. Value of OrderIndex property can be changed automatically by setting value for the Marker or MarkerIndex property.

Options dgMarkerAutoSwitch and dgMarkerAutoToggle included in Options are allowed when the OrderIndex property is changed.

See also: TXColumnTitle.Marker, TXColumnTitle.MarkerIndex, TXCustomDBGrid.OrderFields

TXColumnTitle.PopupMenu

Identifies the pop-up menu associated with the title of column.

```
property PopupMenu: TPopupMenu;
```

Description
Assign a value to PopupMenu to make a pop-up menu appear when the user clicks the right mouse button on the title of column. If the TPopupMenu's AutoPopup property is True, the pop-up menu appears automatically. If the menu's AutoPopup property is False, display the menu with a call to its Popup method.

See also: TColumn.PopupMenu, TXColumnTotal.PopupMenu, TXColumnTitle.DropDownMenu, TXCustomDBGrid.FillerPopupMenu, TXCustomDBGrid.IndicatorPopupMenu

TXColumnTitle.VAlignment

Specifies how text is vertical aligned within the title of column.

```
property VAlignment: TVAlignment;
```

Description
Use VAlignment to specify whether the text is top-justified, bottom-justified, or centered.

See also: TXColumn.VAlignment, TColumnTitle.Alignment

TXColumnTitle.WordWrap

Specifies whether the Caption wraps when it is too long for the width of the column.

```
property WordWrap: Boolean;
```

Description
Set WordWrap to True to allow the Caption to display multiple line of text. When WordWrap is True, text that is too wide for the column wraps at the right margin and continues in additional lines.

Set WordWrap to False to limit the Caption to a single line. When WordWrap is False, text that is too wide for the column appears truncated. See also Ellipsis property.

See also: TXColumn.WordWrap, TXColumnTotal.WordWrap, TXColumnTitle.Ellipsis

TXColumnTitle.Create

Creates and initializes a column title.

```
constructor Create(Column: TXColumn);
```

Description
Call Create to instantiate an instance of TXColumnTitle. Create takes a TXColumn instance as its argument.

Most applications need not create column title instances as these are instantiated by the column.

See also: TXColumn

TXColumnTitle.HeaderLeft

Finds a column on the left side merged by the header.

```
function HeaderLeft(Index: Integer): TXColumn;
```

Description
Call HeaderLeft to retrieve column on the left side merged by header row indicated by Index parameter. Headers for adjacent columns are merged while rows contain identical text.

See also: TXColumnTitle.HeaderRight, TXColumnTitle.Header

TXColumnTitle.HeaderRight

Finds a column on the right side merged by the header.

```
function HeaderRight(Index: Integer): TXColumn;
```

Description
Call HeaderRight to retrieve column on the right side merged by header row indicated by Index parameter. Headers for adjacent columns are merged while rows contain identical text.

See also: TXColumnTitle.HeaderLeft, TXColumnTitle.Header

TXColumnTitle.ToggleMarker

Changes the state of the sorting markers.

```
procedure ToggleMarker(NewOrder: Boolean);
```

Description
Call ToggleMarker to change state of the sorting Marker programmicaly. When NewOrder is True, markers in all other columns are clear.

If Options includes dgMarkerAutoToggle and AutoToggle property is True, ToggleMarker method is called internaly when the user clicks on the title of column. In other way you may call ToggleMarker in OnTitleClick event handler to change state of the sorting marker using default changing scheme. When you click on a title of column with empty FieldName, the default order is restored (no visible markers).

Default using is: `ToggleMarker(not (ssCtrl in LastShiftState))`

See also: TXColumnTitle.AutoToggle, TXColumnTitle.Marker, TXCustomDBGrid.Options, TCustomDBGrid.OnTitleClick

TXColumnTotal

TXColumnTotal represents the base total cell of a data grid column (TXColumn).

Unit
XDBGrids

Description
TXDBGrid uses a TXDBGridColumns to maintain a collection of TXColumn objects. Each TXColumn can have 4 kinds of totals cells (TotalHeader, TotalValues, TotalFields, TotalFooter). Each total cell is derived from TXColumnTotal base class.

TXDBGrid has also an associated Totals object that holds common properties for all totals cells.

See also: TXColumnTotalValue, TXColumnTotalFields, TXColumnTotalHeader, TXColumnTotalFooter

TXColumnTotal.Alignment

Specifies how text is aligned within the total cell.

```
property Alignment: TAlignment;
```

Description
Use Alignment to specify whether the total cell is left-justified, right-justified, or centered. See also DefaultAlignment.

See also: TColumn.Alignment, TXColumnTotal.VAlignment, TXColumnTotal.DefaultAlignment

TXColumnTotal.Caption

Specifies the text that appears in the total cell.

```
property Caption: string;
```

Description
The Caption property contains a text string that is displayed in total cell. When TotalResult is trCaption the Caption property holds static text string that is displayed in total cell. You can set different text for dgCalcWholeDataSet and dgCalcSelectedRows modes separeted by "|" char (e.g. "All record(s)|Selected record(s)"). When "|" char is omitted the Caption property holds common text for both modes. See also SelectedRows property.

When TotalResult is trCalcValue the Caption property holds formated Value as text string that is displayed in total cell. Each time when Value property is changed (recalculated) the Caption property is modified with using FormatValue function. When TotalResult is trCalcValue the Caption property always holds text for both modes. See also CalcApply property.

See also: TColumnTitle.Caption, TXColumnTotal.DisplayText, TXColumnTotal.TotalResult

TXColumnTotal.Color

Specifies the background color for the total cell.

```
property Color: TColor;
```

Description
The Color property determines the background color of the total cell. You can set Color to one of the constants defined in the Graphics unit (such as clBlue), or to an explicit RGB integer value. The default value is determined by DefaultColor.

See also: TXColumnTotal.DefaultColor, TColumnTitle.Color, TXDBGridTotals.Color

TXColumnTotal.Column

Specifies the TColumn object that is associated with the total cell.

```
property Column: TXColumn;
```

Description
Use Column to determine the column associated with the total cell.

See also: TXColumn.TotalValues, TXColumn.TotalFooter, TXColumn.TotalHeader, TXColumn.TotalFields

TXColumnTotal.DisplayText

Represents the total's value as it is displayed in total cell.

```
property DisplayText: string;
```

Description
DisplayText is a read-only string representation of a total's value for displaying in total cell. It represents the Caption, Field.DisplayText or column tkDisplay text depend on current value of TotalResult property. See also OnTotalGetText event handler.

See also: TXColumnTotal.Caption, TXColumnTotal.TotalResult

TXColumnTotal.Ellipsis

Specifies whether the Caption is completed by ... when it is too long for the width of the column.

```
property Ellipsis: Boolean;
```

Description
Set Ellipsis to True to allow the Caption completed by triple dot. When Ellipsis is True, text that is too wide for the column is completed by triple dot.

When Ellipsis is False, text that is too wide for the column appears truncated. See also WordWrap property.

See also: TXColumnTitle.Ellipsis, TXColumn.Ellipsis, TXColumnTotal.WordWrap

TXColumnTotal.Field

Indicates the TField instance represented by the total cell.

```
property Field: TField;
```

Description
Field points to the TField object that corresponds to the dataset field displayed in the total cell. The value of Field will be nil (Delphi) or NULL (C++), if the FieldName does not correspond to a field in the dataset. If FieldName is empty and TotalResult is trDataField or trCalcValue the Field property points to Column.Field object.

See also: TXColumnTotal.FieldName, TColumn.Field

TXColumnTotal.FieldName

Indicates the name of the field that appears in total cell.

```
property FieldName: string;
property FieldName: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
Setting FieldName changes the Field property so that it points to the dataset field with the same name. If the dataset does not have a field with the same name, Field is set to nil (Delphi), or NULL (C++). You can select field from both DataSource and Totals.DataSource depend on value of TotalResult property. See also TotalRecNo property.

See also: TXColumn.FieldName, TXColumnTotal.Field, TXColumnTotalValue.TotalRecNo

TXColumnTotal.Font

Controls the font in which the total value is displayed.

```
property Font: TFont;
```

Description
The Font property points to a TFont object that determines typographic attributes of text displayed in the total cell. The default value is determined by DefaultFont.

See also: TColumn.Font, TXColumnTotal.DefaultFont, TXDBGridTotals.Font

TXColumnTotal.Hint

Specifies the text string that can appear when the user moves the mouse pointer over a total cell.

```
property Hint: string;
```

Description
Set Hint to a string that provides more information about the meaning of the total value. The hint text appears in a Help Hint window when the user pauses with the mouse over the total cell if HintOptions includes hoShowTotalHints.

See also: TXColumnTitle.Hint, TXColumn.Hint

TXColumnTotal.ImageIndex

Indicates which image maintained by the Totals.Images appears to the left side of title's caption.

```
property ImageIndex: Integer;
```

Description
Set ImageIndex to designate an image that should appear to the left side of total's value. The ImageIndex specifies a zero-offset index into the Totals.Images property.

See also: TXColumnTitle.ImageIndex, TXDBGridTotals.Images

TXColumnTotal.PopupMenu

Identifies the pop-up menu associated with the total cell.

```
property PopupMenu: TPopupMenu;
```

Description
Assign a value to PopupMenu to make a pop-up menu appear when the user clicks the right mouse button on the total cell. If the TPopupMenu's AutoPopup property is True, the pop-up menu appears automatically. If the menu's AutoPopup property is False, display the menu with a call to its Popup method.

See also: TXColumnTitle.PopupMenu, TColumn.PopupMenu

TXColumnTotal.ToolTips

Specifies whether the total cell have tooltips.

```
property ToolTips: Boolean;
```

Description
Set ToolTips to True to specify that total cell have tooltips (Help Hints). See also HintOptions.

See also: TXColumn.ToolTips, TXColumnTotal.ToolTipsWidth

TXColumnTotal.ToolTipsWidth

Specifies how the width of tooltips window is calculated.

```
property ToolTipsWidth: TToolTipsWidth;
```

Description
Set the ToolTipsWidth property to determine width of tooltips window. These are the possible values of ToolTipsWidth:

Value	Meaning
ttAutoWidth	The width of tooltips window is calculated according to width of total's text.
ttFixedWidth	The width of tooltips window is calculated according to width of column.

See also: TXColumnTotal.ToolTips

TXColumnTotal.TotalBox

Specifies whether the total cell has a box.

```
type
  TTotalBox = (tbFalse, tbTrue, tbAuto);
property TotalBox: TTotalBox;
```

Description
Use TotalBox to appear box in total cell. These are the possible values of TotalBox:

Value	Meaning
tbFalse	The box never appears in the total cell.
tbTrue	The box always appears in the total cell.
tbAuto	The box appears in the total cell depend on OptionsEx property value.

See also: TXDBGridTotals.BoxStyle

TXColumnTotal.TotalResult

Specifies what result apperas within the total cell.

```
type
  TTotalResult = (trNone, trCaption, trDataField, trTotalField, trCalcValue);
property TotalResult: TTotalResult;
```

Description
Use TotalResult to specify what result should be showing within the total cell. These are the possible values of TotalResult:

Value	Meaning
trNone	No value appears within total cell.
trCaption	Static text string stored in Caption property appears within total cell.
trDataField	If FieldName is not empty the Field.DisplayText value appears within total cell. If FieldName is empty the text displayed in data cell of current Column appears within total cell (See also: ColGetText function with parameter tkDisplay). The FieldName property must determine Field from main DataSource.DataSet.
trTotalField	If FieldName is not empty the Field.DisplayText value appears within total cell. If FieldName is empty no value appears within total cell. The FieldName property must determine Field from Totals.DataSource.DataSet. See also TotalRecNo property.
trCalcValue	The result of CalcValue function stored into Value variant property and formated with using FormatValue function to the Caption string property appears within total cell. Each time when Value property is changed (recalculated) the Caption property is modified with using FormatValue function. The FieldName property must determine Field from main DataSource.DataSet. If FieldName is empty the Field points to Column.Field property. Each CalcValue function always operate on Field.Value variant value. See also OnTotalCalcValue event.

See also DisplayText read-only property.

See also: TXColumnTotal.Caption, TXColumnTotal.DisplayText, TXColumnTotal.FieldName, TXColumnTotalValue.CalcValue

TXColumnTotal.VAlignment

Specifies how text is vertical aligned within the total cell.

```
property VAlignment: TVAlignment;
```

Description
Use VAlignment to specify whether the text is top-justified, bottom-justified, or centered.

See also: TXColumn.VAlignment, TXColumnTotal.Alignment

TXColumnTotal.Visible

Specifies whether the total value is visible in total cell.

```
property Visible: Boolean;
```

Description
Use Visible to appear total value in total cell. The default value is True. When you want to have an invisible calculated value you can set Visible property to False.

See also: TXCustomDBGrid.OptionsEx

TXColumnTotal.WordWrap

Specifies whether the Caption wraps when it is too long for the width of the column.

```
property WordWrap: Boolean;
```

Description
Set WordWrap to True to allow the Caption to display multiple line of text. When WordWrap is True, text that is too wide for the column wraps at the right margin and continues in additional lines.

Set WordWrap to False to limit the Caption to a single line. When WordWrap is False, text that is too wide for the column appears truncated. See also Ellipsis property.

See also: TXColumnTitle.WordWrap, TXColumn.WordWrap, TXColumnTotal.Ellipsis

TXColumnTotal.Assign

Copies the contents of the source column total to a new column total.

```
procedure Assign(Source: TPersistent); override;
```

Description
Assign copies the published properties of the source object if it is another TXColumnTotal object. Otherwise, it calls the inherited method, which allows properties to be copied from any object that implements its AssignTo method with a target of TXColumnTotal.

See also: TXColumnTotalValue.Assign

TXColumnTotal.Create

Creates and initializes a TXColumnTotal object.

```
constructor Create(Collection: TCollection); override;
```

Description
Call Create to instantiate a TXColumnTotal object. Create should take a TXColumnTotalValues instance as its argument.

See also: TXColumnTotal.Destroy

TXColumnTotal.DefaultAlignment

Returns the default alignment of the total cell.

```
property DefaultAlignment: TAlignment;
```

Description
DefaultAlignment returns the alignment for the total cell that should be used if the Alignment property is not explicitly set. The DefaultAlignment property for TXColumnTotal always returns the Column.Alignment. Descendants of TXColumnTotal can override this to compute the default alignment in an appropriate way.

See also: TXColumnTotal.Alignment

TXColumnTotal.DefaultColor

Returns the default background color for the column total.

```
property DefaultColor: TColor;
```

Description
Use DefaultColor to determine the background color of the total cell if the Color property is not explicitly set. DefaultColor returns the Totals.Color of the data grid to which the column belongs. If the column has no associated data grid, DefaultColor returns clBtnFace.

See also: TXColumnTotal.Color

TXColumnTotal.DefaultFont

Returns the default font for the total value.

```
property DefaultFont: TFont;
```

Description
Use DefaultFont to determine the font of the total value if the Font property is not explicitly set. DefaultFont returns the Totals.Font of the data grid to which the column belongs. If the column has no associated data grid, DefaultFont returns the current value of the associated column's Font property.

See also: TXColumnTotal.Font

TXColumnTotal.Destroy

Destroys the column total and frees its memory.

```
destructor Destroy; override;
```

Description
Destroy eliminates the TColumnTotal instance along with its associated objects.

See also: TXColumnTotal.Create

TXColumnTotal.RestoreDefaults

Restores the column total's default settings.

```
procedure RestoreDefaults; virtual;
```

Description
RestoreDefaults reinitializes layout of column total cell.

See also: TColumn.RestoreDefaults

TXColumnTotalValue

TXColumnTotalValue represents the base total cell with values calculated in TXDBGrid.

Unit
XDBGrids

Description
TXDBGrid uses a TXDBGridColumns to maintain a collection of TXColumn objects. Each TXColumn can have 3 kinds of totals cells with values calculated in TXDBGrid (TotalHeader, TotalValues, TotalFooter). Each total cell with values calculated in TXDBGrid is derived from TXColumnTotalValue base class.

TXDBGrid has also an associated Totals object that holds common properties for all totals cells.

See also: TXColumnTotal, TXColumnTotalValues, TXColumnTotalHeader, TXColumnTotalFooter

TXColumnTotalValue.CalcApply

Specifies when the calculated value should be processed.

```
type
  TCalcApply = (caWholeDataSet, caSelectedRows, caBoth);
property CalcApply: TCalcApply;
```

Description
Use CalcApply to determine when the calculated value should be processed. These are the possible values of CalcApply:

Value	Meaning
caWholeDataSet	Calculate total value only when whole dataset is processed.
caSelectedRows	Calculate total value only when selected rows are processed.
caBoth	Calculate total value when whole dataset or selected rows are processed.

See also CalcValue property.

See also: TXDBGridTotals.SelectedRows, TXColumnTotalValue.CalcValue

TXColumnTotalValue.CalcValue

Specifies type of calculated value that appears within the total cell.

```
type
  TCalcValue = (cvSum, cvAvg, cvCount, cvMax, cvMin, cvFirst, cvLast,
    cvUnique, cvCustom);
property CalcValue: TCalcValue;
```

Description
Specify CalcValue to determine type of calculated value that appears in the total cell. These are the possible values of CalcValue:

Value	Meaning
cvSum	Sum of Field.Value appears within total cell. The field must be numeric.
cvAvg	Average of Field.Value appears within the total cell. The field must be numeric.
cvCount	Count of Field.Value appears within the total cell.
cvMax	Maximum of Field.Value appears the within total cell.

cvMin	Minimum of Field.Value appears within the total cell.
cvFirst	First Field.Value according to current order appears within the total cell.
cvLast	Last Field.Value according to current order appears within the total cell.
cvUnique	If all calculated values are identical the value appears within the total cell. Otherwise, the result is Null.
cvCustom	By default, the result is Unassigned. You can write OnTotalCalcValue event handler and return any value to apper within the total cell.

See also NullValue property.

See also: TXColumnTotalValue.NullValue, TXCustomDBGrid.OnTotalCalcValue

TXColumnTotalValue.Counter

Holds counter during calculations.

```
property Counter: Integer;
```

Description
Use Counter in OnTotalCalcValue event handler to determine number of processed field's values (number of records) during CalcValue function is performed. The Counter is 0 before cpReset pass and it's incremented during each cpCalcRec pass for each not Null value if NullValue property is nvExclude or for any value in other cases. The final result determines number of processed values and it's valid during cpFinal pass.

See also: TXColumnTotalValue.TempValue, TXColumnTotalValue.Value

TXColumnTotalValue.Format

Specifies format string for total cell value.

```
property Format: string;
```

Description
Use Format property to determine format string for Value. Each time the Value property is changed FormatValue function converts Value to it's string representation and strores formatted value to Caption property.

When Format is empty the default formatting for each type of Value will be performed. When Format is not empty it must contain format string (any text) with one argument specifier. You can set different format for dgCalcWholeDataSet and dgCalcSelectedRows modes separeted by "|" char (e.g. "All record(s): %d|Selected record(s): %d"). When "|" char is omitted the Format property holds common format for both modes. See also SelectedRows property.

The argument specifier must be appropriate to expected type of Value and according to: SysUtils.Format function for strings, numeric and currency values, FormatDateTime function for date & time values, CheckBoxValues format for Boolean values. When the format's argument specifier is not appropriate for current type of Value, format string will be ignored and default formatting will be performed (in design-time you will hear a beep). To use Float format strings for ordinal values (e.g %m for Integer) you should set True for FormatAsFloat property.

Use property editor to select one of most popular format specifier for expected type of Value. These are the most popular format specifiers:

Format	Type	Result	
%True;False%	Boolean	(True)	
%Yes;No%	Boolean	(Yes)	
%On;Off%	Boolean	(On)	
%1;0%	Boolean	(1)	
%c%	Date	Date&Time	
%ddddd%	Date	ShortDate	
%dddddd%	Date	LongDate	
%t%	Date	ShortTime	
%tt%	Date	LongTime	
%ddd%	Date	ShortDay	
%dddd%	Date	LongDay	
%mmm%	Date	ShortMonth	
%mmmm%	Date	LongMonth	
%dd%	Date	Day	
%mm%	Date	Month	
%yyyy%	Date	Year	
%hh%	Date	Hour	
%nn%	Date	Minute	
%ss%	Date	Second	
%zzz%	Date	Milisecond	
%d	Ordinal	Decimal	(1000)
%u	Ordinal	Unsigned	(1000)
%x	Ordinal	Hexadecimal	(3E8)
%8d	Ordinal	Decimal	(1000)
%8u	Ordinal	Unsigned	(1000)
%8x	Ordinal	Hexadecimal	(3E8)
%.8d	Ordinal	Decimal	(00001000)
%.8u	Ordinal	Unsigned	(00001000)
%.8x	Ordinal	Hexadecimal	(000003E8)
%.0f	Float	Fixed	(1000)
%.0n	Float	Number	(1 000)
%.0m	Float	Money	(1 000 $)
%.0n%%	Float	Percent	(1 000%)
%f	Float	Fixed	(1000,10)
%g	Float	General	(1000,1)
%n	Float	Number	(1 000,10)
%m	Float	Money	(1 000,10 $)
%e	Float	Scientific	(1,00010000000000E+003)

%n%%	Float	Percent	(1 000,10%)
%.4f	Float	Fixed	(1000,1000)
%.4n	Float	Number	(1 000,1000)
%.4m	Float	Money	(1 000,1000 $)
%.4e	Float	Scientific	(1,000E+003)
%.4n%%	Float	Percent	(1 000,1000%)
%12f	Float	Fixed	(1000,10)
%12g	Float	General	(1 000,1)
%12n	Float	Number	(1 000,10)
%12m	Float	Money	(1 000,10 $)
%12n%%	Float	Percent	(1 000,10%)
%s	String	(Text)	
%8s	String	(Text)	

See also: TXColumnTotalValue.FormatAsFloat, TXColumnTotalValue.Value, TXColumnTotalValue.Caption, TXColumnTotalValue.FormatValue

TXColumnTotalValue.FormatAsFloat

Determines type of formatted total cell's value.

```
property FormatAsFloat: Boolean;
```

Description
Use FormatAsFloat to determine using Float format for non-float type of Value. Set True for non-float type of Value to force using float Format string. When FormatAsFloat is True, FormatValue function converts Value to Float type before the Value will be formatted. It's especially useful to format ordinal values as money.

See also: TXColumnTotalValue.Format, TXColumnTotalValue.Value

TXColumnTotalValue.NullValue

Specifies how the Null value is processing during calculation.

```
type
  TNullValue = (nvExclude, nvInclude, nvAsZero);
property NullValue: TNullValue;
```

Description
Use NullValue to determine how the Null value stored in a field should be processed by CalcValue function. These are the possible values of NullValue:

Value	Meaning
nvExclude	Null values are excluded during calculations.
nvInclude	Null values are respected like any other value.
nvAsZero	Null values are converted to 0.

Notice. The result of any CalcValue function may be different depend of state of NullValue. By default, NullValue is set to nvExclude.

See also: TXColumnTotalValue.CalcValue

TXColumnTotalValue.TempValue

Holds temporary total value during calculations.

```
property TempValue: Variant;
```

Description
Use TempValue in OnTotalCalcValue event handler to determine current state of calculation during CalcValue function is performed. The TempValue is Unassigned before cpReset pass, it's modified according to CalcValue function during each cpCalcRec pass and final result is stored to Value property after cpFinal pass.

See also: TXColumnTotalValue.Counter, TXColumnTotalValue.Value

TXColumnTotalValue.TotalRecNo

Indicates the record in the total dataset.

```
property TotalRecNo: Integer;
```

Description
Specify TotalRecNo when TotalResult is trTotalField to select record in the Totals.DataSource.DataSet. You can specify TotalRecNo property for each DataSet starting from 1 even if RecNo property is not supported by DataSet descendant. Setting FieldName and TotalRecNo you can treat Totals.DataSource.DataSet as two-dimesional array with totals values.

See also: TXDBGridTotals.DataSource, TXDBGridTotals.DataMaxRow, TXColumnTotal.TotalResult

TXColumnTotalValue.Value

Specifies the value that appears in the total cell.

```
property Value: Variant;
```

Description
The Value property contains a variant value that is formatted to the Caption string value and displayed in total cell. When TotalResult is trCaption the Value property can store static variant value that will be automatically formatted to the Caption string value with using FormatValue function.

When TotalResult is trCalcValue the Value property holds result of CalcValue function. Each time when Value property is changed (recalculated) the Caption property is modified with using FormatValue function.

See also: TXColumnTotalValue.Format, TXColumnTotal.Caption, TXColumnTotalValue.FormatValue

TXColumnTotalValue.Assign

Copies the contents of the source column total value to a new column total value.

```
procedure Assign(Source: TPersistent); override;
```

Description
Assign copies the published properties of the source object if it is another TXColumnTotalValue object. Otherwise, it calls the inherited method, which allows properties to be copied from any object that implements its AssignTo method with a target of TXColumnTotalValue.

See also: TXColumnTotal.Assign

TXColumnTotalValue.Create

Creates and initializes a TXColumnTotalValue object.

```
constructor Create(Collection: TCollection); override;
```

Description
Call Create to instantiate a TXColumnTotalValue object. Create should take a TXColumnTotalValues instance as its argument.

See also: TXColumnTotal.Create

TXColumnTotalValue.FormatValue

Formats variant value to it's string representation.

```
function FormatValue(AValue: Variant): string; virtual;
```

Description
Use FormatValue to format value of any type according to total cell's Format and FormatAsFloat property. FormatValue is calling automatically when Value property is chanaged and result of this function is stored to Caption property to appear Value within total cell.

See also: TXColumnTotalValue.Format, TXColumnTotalValue.Value

TXColumnTotalValues

TXColumnTotalValues represents a collection of total cells with calculated values for data grid column (TXColumn).

Unit
XDBGrids

Description
TXDBGrid uses a TXDBGridColumns to maintain a collection of TXColumn objects. Each TXColumn can have 4 kinds of totals cells (TotalHeader, TotalValues, TotalFields, TotalFooter). TXColumnTotalValues represents a collection of total cells with calculated values for a column. TXColumnTotalValues collection is a container for TXColumnTotalValue objects.

TXDBGrid has also an associated Totals object that holds common properties for all totals cells.

See also: TXColumnTotalValue, TXColumnTotalFooter, TXColumnTotalHeader, TXColumn.TotalValues

TXColumnTotalValues.Column

Specifies the TXColumn object that is associated with the column TotalValues collection.

```
property Column: TXColumn;
```

Description
Use Column to determine the column associated with the TotalValues collection.

See also: TXColumn.TotalValues

TXColumnTotalValues.Items

Lists the total values in the collection.

```
property Items[Index: Integer]: TXColumnTotalValue;
```

Description
Use Items to access individual total values. The value of the Index parameter represents the position of the total value in the TotalValues collection.

Items is the default property of TXColumnTotalValues. This means that the property name, Items, can be omitted when referring to values of the TXColumnTotalValues object. Thus, the line

FirstValue := XDBGrid1.Columns[0].TotalValues.Items[0];

can also be written

FirstValue := XDBGrid1.Columns[0].TotalValues[0];

See also: TXColumnTotalValue

TXColumnTotalValues.Add

Creates a new TXColumnTotalValue instance and adds it to the Items array.

```
function Add: TXColumnTotalValue;
```

Description
Add returns the new total value. At design time, use the collection's TotalValues editor to add new total value to the collection.

See also: TXColumnTotalValue

TXColumnTotalValues.Create

Creates and initializes a TotalValues object.

```
type
  TXColumnTotalValueClass = class of TXColumnTotalValue;
constructor Create(Column: TXColumn; ValueClass: TXColumnTotalValueClass);
override;
```

Description
Call Create to instantiate an instance of TXColumnTotalValues. The Create method takes two parameters: a TXColumn instance object and TXColumnTotalValue class (or the name of a class derived from TXColumnTotalValue). You not need to create this object directly. This is instantiated by each grid's column.

See also: TXColumnTotalValue

TXColumnTotalValues.GetEnumerator

Creates enumerator for TXColumnTotalValues class.

```
function GetEnumerator: TXDBGridColumnsEnumerator; {* ver. 5.6 *} // For
Delphi 2009 or higher
```

Description
You not need to call this function directly. It allows you to iterate over all values in the column total.

```
type
  ColumnTotalValue: TXColumnTotalValue;
begin
  for ColumnTotalValue in XDBGrid1[ColumnName].TotalValues do
  begin
    // Perform action on each total value in XDBGrid1[ColumnName]
  end;
end;
```

See also: TXDBGridTotals.GetEnumerator

TXColumnTotalHeader

TXColumnTotalHeader represents the total cell with calculated value for first total row in data grid column (TXColumn).

Unit
XDBGrids

Description
TXDBGrid uses a TXDBGridColumns to maintain a collection of TXColumn objects. Each TXColumn can have 4 kinds of totals cells (TotalHeader, TotalValues, TotalFields, TotalFooter). TXColumnTotalHeader represents the first of total cells for a column. TXColumnTotalHeader is derived from TXColumnTotalValue base class.

TXDBGrid has also an associated Totals object that holds common properties for all totals cells.

See also: TXColumnTotalValue, TXColumnTotalFooter, TXColumnTotalValues, TXColumn.TotalHeader

TXColumnTotalHeader.Create

Creates and initializes a TotalHeader object.

```
constructor Create(Column: TXColumn); reintroduce;
```

Description
Call Create to instantiate an instance of TXColumnTotalHeader. Create takes a TXCustom instance as its argument. You not need to create this object directly. This is instantiated by each grid's column.

See also: TXColumnTotal.Create

TXColumnTotalFooter

TXColumnTotalFooter represents the total cell with calculated value for last total row in data grid column (TXColumn).

Unit
XDBGrids

Description
TXDBGrid uses a TXDBGridColumns to maintain a collection of TXColumn objects. Each TXColumn can have 4 kinds of totals cells (TotalHeader, TotalValues, TotalFields, TotalFooter). TXColumnTotalFooter represents the last of total cells for a column. TXColumnTotalFooter is derived from TXColumnTotalValue base class.

TXDBGrid has also an associated Totals object that holds common properties for all totals cells.

See also: TXColumnTotalValue, TXColumnTotalHeader, TXColumnTotalValues, TXColumn.TotalFooter

TXColumnTotalFooter.Create

Creates and initializes a TotalFooter object.

```
constructor Create(Column: TXColumn); reintroduce;
```

Description
Call Create to instantiate an instance of TXColumnTotalFooter. Create takes a TXCustom instance as its argument. You not need to create this object directly. This is instantiated by each grid's column.

See also: TXColumnTotal.Create

TXColumnTotalFields

TXColumnTotalFields represents the total cells with fields values retrieved from Totals.DataSource.

Unit
XDBGrids

Description
TXDBGrid uses a TXDBGridColumns to maintain a collection of TXColumn objects. Each TXColumn can have 4 kinds of totals cells (TotalHeader, TotalValues, TotalFields, TotalFooter). TXColumnTotalFields represents the total cells with field's values retrieved from Totals.DataSource. TXColumnTotalFields is derived from TXColumnTotal base class.

TXDBGrid has also an associated Totals object that holds common properties for all totals cells.

See also: TXColumnTotal, TXColumn.TotalFields

TXColumnTotalFields.Create

Creates and initializes a TotalFields object.

```
constructor Create(Column: TXColumn); reintroduce;
```

Description
Call Create to instantiate an instance of TXColumnTotalFields. Create takes a TXCustom instance as its argument. You not need to create this object directly. This is instantiated by each grid's column.

See also: TXColumnTotal.Create

TXDBGridTotals

TXDBGridTotals holds common properties for totals cells in TXDBGrid.

Unit
XDBGrids

Description
TXDBGrid uses a TXDBGridColumns to maintain a collection of TXColumn objects. Each TXColumn can have 4 kinds of totals cells (TotalHeader, TotalValues, TotalFields, TotalFooter). TXDBGrid has an associated Totals object that holds common properties for all totals cells.

See also: TXCustomDBGrid.Totals, TXColumnTotalFields, TXColumnTotalHeader, TXColumnTotalValues, TXColumn.TotalFooter

TXDBGridTotals.Active

Specifies whether any total row may appear in the grid.

```
property Active: Boolean; {* ver. 5.0 *}
```

Description
Use the Active property to control the visibility of total rows at runtime. If Active is true, one or more total rows may appears in the grid (is Visible or AutoHidden). If Active is false, no total rows is active. See also dgSelectedAutoHidden option in OptionsEx property.

Active is a read-only property.

See also: TXCustomDBGrid.OptionsEx

TXDBGridTotals.BoxStyle

Specifies style of total box drawn in the total cell.

```
type
  TTotalBoxStype = TCheckBoxStyle;
property BoxStyle: TTotalBoxStyle;
```

Description
Use BoxStyle property to select a style of total box drawn in the total cell. The BoxStyle property is practicable only for non-Windows XP styles. See also OptionsEx and TotalBox property to read when the total box is drawn in total cell.

See also: TXColumnTotal.TotalBox

TXDBGridTotals.CalcValues

Determines when the total values are calculated.

```
property CalcValues: Boolean;
```

Description
Set CalcValues to control when the totals cells with CalcValue are calculated. This property is default set to True. It means that all totals cells with TotalResult=trCalcValue are calculated by the grid when it's needed.

You may want to stop calculation when you modify in code several fields in DataSet linked to the grid without using DisablePosition/EnablePosition or DisableControls/EnableControls statements. Then you should set CalcValues switch to False.

When CalcValues switch is False each total cell remembers the last calculated value and this value is still displayed in the total cell. When you restore CalcValues to True you should call UpdateTotals to recalculate all totals cells. When EnablePosition or EnableControls will be applied all calculated values are automatically recalculated.

See also: TXCustomDBGrid.UpdateTotals, TXColumnTotalValue.CalcValue, TXColumnTotal.TotalResult

TXDBGridTotals.Color

Specifies the background color of the total rows in the grid.

```
property Color: TColor;
```

Description
Set Color to specify the color for the total cells in the grid.

See also: TXColumnTotal.Color

TXDBGridTotals.DataMaxRow

Specifies maximum number of records from total dataset to display in total rows.

```
property DataMaxRow: Integer;
```

Description
Use DataMaxRow to prevent loading large total DataSource.DataSet into memory. Default value for DataMaxRow is set to 5. You can increase this property to reasonable value or set DataMaxRow to 0 to lift this limit, but you should remember that this functionality works successfully for reasonable number of records in total dataset.

See also: TXDBGridTotals.DataSource, TXColumnTotalValue.TotalRecNo

TXDBGridTotals.DataSource

Identifies the link to the totals dataset where the grid finds its totals data.

```
property DataSource: TDataSource;
```

Description
Set DataSource to the TDataSource object that links to the totals dataset where the grid should fetch its totals data. This DataSource allows the grid to read totals data from the dataset. To show totals data from dataset in grid's totals rows you should include dgTotalFields in OptionsEx property and select FieldName for selected column's TotalFields properties. When dgTotalFields option is included in OptionsEx the grid create one total row for one record in total dataset. It's especially useful to show totals values calculated by GROUP BY clause in SQL statement.

This functionality works successfully for reasonable number of records in total dataset. Be careful to not open large dataset as total dataset. To prevent loading large dataset into memory DataMaxRow property is default set to 5. Remember that whole dataset must reside in memory to successfully use TotalFields.

You can also show selected fields from selected records from this dataset into any other kind of total rows (TotalHeader, TotalValues, TotalFooter). To do this you should set TotalResult property to trTotalField, FieldName property to selected field and TotalRecNo to RecNo in total dataset (TotalRecNo property can be used for each DataSet starting from 1 even if RecNo property is not supported by DataSet descendant). In this way you can use any dataset as two-dimensional array of totals values. It's esspecially useful for using a memory table which contains totals values.

See also: TXDBGridTotals.DataMaxRow, TXColumnTotalValue.TotalRecNo

TXDBGridTotals.Font

Describes the font used to draw the column totals in the grid.

```
property Font: TFont;
```

Description
Set Font to change the font used to draw the column total values. Font is only meaningful if the OptionsEx property includes dgTotalHeader, dgTotalValues, dgTotalFields or dgTotalFooter.

See also: TXColumnTotal.Font

TXDBGridTotals.Grid

Indicates the grid that contains this totals.

```
property Grid: TXCustomDBGrid;
```

Description
Use the Grid property to access the grid that contains this totals.

See also: TXCustomDBGrid

TXDBGridTotals.Images

Determines which image list is associated with the totals.

```
property Images: TImageList;
```

Description
Use Images to provide a customized list of bitmaps that can be displayed to the left of a total's values. Individual total cells specify the image from this list that should appear by setting their ImageIndex property.

See also: TXColumnTotal.ImageIndex, TXCustomDBGrid.TitleImages

TXDBGridTotals.LinesCount

Specifies number of lines for multi-line totals of columns in the grid.

```
property LinesCount: Integer;
```

Description
Set LinesCount to specify number of lines for multi-line totals of columns in the grid. LinesCount is only meaningful if the OptionsEx property includes dgTotalHeader, dgTotalValues, dgTotalFields or dgTotalFooter.

See also: TXCustomDBGrid.LinesCount, TXCustomDBGrid.TitleLinesCount

TXDBGridTotals.Selected

Specifies when SelectedRows list or SelectedCols list is not empty.

```
property Selected: Boolean; {* ver. 4.3 *}
```

Description
Use Selected to determine when totals selected rows have calculated values for rows, columns or cells selected in the grid. The Selected property returns True when SelectedRows or SelectedCols returns True. See also: MultiSelect.

Selected is a read-only property

See also: TXDBGridTotals.SelectedCols, TXDBGridTotals.SelectedRows, TXColumnTotalValue.CalcApply

TXDBGridTotals.SelectedColor

Returns the background color for the totals selected rows.

```
property SelectedColor: TColor;
```

Description
Use SelectedColor to determine the background color of the totals selected rows if the SelectedRows property is True. When IsGridThemed is True SelectedColor determine the foreground color of the totals selected rows. When SelectedColor property is set to clDefault the DefaultSelectedColor will be used. When SelectedColor property is set to clNone the color of totals selected rows will not be changed.

See also: TXDBGridTotals.SelectedRows, TXDBGridTotals.DefaultSelectedColor

TXDBGridTotals.SelectedCols

Specifies when SelectedCols list is not empty.

```
property SelectedCols: Boolean; {* ver. 4.3 *}
```

Description
Use SelectedCols to determine when totals selected rows have calculated values for SelectedCols list. TXDBGrid can calculate totals value for whole DataSet or either for SelectedRows/SelectedCols list only. When all (dgCalcSelectedRows) or selected (dgTotalHeaderSelected, dgTotalValuesSelected, dgTotalFooterSelected) totals rows contain totals values calculated for SelectedCols list only the totals selected rows have changed background or foreground Color to SelectedColor.

SelectedCols a read-only property

See also: TXDBGridTotals.Selected, TXDBGridTotals.SelectedRows, TXColumnTotalValue.CalcApply

TXDBGridTotals.SelectedRows

Specifies when SelectedRows list is not empty.

```
property SelectedRows: Boolean;
```

Description
Use SelectedRows to determine when totals selected rows have calculated values for SelectedRows ist. TXDBGrid can calculate totals value for whole DataSet or either for SelectedRows/SelectedCols list only. When all (dgCalcSelectedRows) or selected (dgTotalHeaderSelected, dgTotalValuesSelected, dgTotalFooterSelected) totals rows contain totals values calculated for SelectedRows only the totals selected rows have changed background or foreground Color to SelectedColor.

SelectedRows is a read-only property

See also: TXDBGridTotals.Selected, TXDBGridTotals.SelectedCols, TXDBGridTotals.SelectedColor, TXColumnTotalValue.CalcApply

TXDBGridTotals.Visible

Specifies whether any total row is visible in the grid.

```
property Visible: Boolean;
```

Description
Use the Visible property to control the visibility of total rows at runtime. If Visible is true, one or more total rows appears in the grid. If Visible is false, no total rows is visibled. See also dgTotalHeader, dgTotalValues, dgTotalFields and dgTotalFooter option for OptionsEx property.

Visible is a read-only property.

See also: TXCustomDBGrid.OptionsEx

TXDBGridTotals.Create

Creates and initializes a Totals object.

```
constructor Create(AGrid: TXCustomDBGrid);
```

Description
Call Create to instantiate an instance of TXDBGridTotals. Create takes a TXCustomDBGrid instance as its argument. You not need to create Totals object directly. This is instantiated by XDBGrid.

See also: TXDBGridTotals.Destroy

TXDBGridTotals.DefaultSelectedColor

Returns the default background color for the totals selected rows.

```
function DefaultSelectedColor: TColor;
```

Description
Use DefaultSelectedColor to determine the default background color of the totals selected rows if the SelectedRows property is True. When IsGridThemed is True DefaultSelectedColor determine the default foreground color of the totals selected rows. DefaultSelectedColor will be used only when SelectedColor property is set to clDefault.

See also: TXDBGridTotals.SelectedColor

TXDBGridTotals.Destroy

Destroys an instance of TXDBGridTotals.

```
destructor Destroy; override;
```

Description
Destroy frees the helper objects of the TXDBGridTotals before destroying the instance. The user never need to call this method directly.

See also: TXDBGridTotals.Create

TXDBGridTotals.GetEnumerator

Creates enumerator for TXDBGridTotals class.

```
function GetEnumerator: TXDBGridTotalsEnumerator; {* ver. 5.6 *} // For Delphi 2009 or higher
```

Description
You not need to call this function directly. It allows you to iterate over all values in the column total.

```
type
  Bookmark: TBookmark;
begin
  for Bookmark in XDBGrid1.Totals do
  begin
    // Perform action on each record into XDBGrid1.Totals.DataSource.DataSet
  end;
end;
```

See also: TXColumnTotalValues.GetEnumerator

TXColumnReport

TXColumnReport represents an extension of a data-grid column (TXColumn) for a report.

Unit
XDBGrids

Description
TXDBGrid uses a TXDBGridColumns to maintain a collection of TXColumn objects. Each TXColumn has an associated TXColumnReport that holds additional information for TXQRGrid. The TXColumnReport instance is stored in the column's Report property.

See also: TXColumn, TXColumn.Report, TXDBGridColumns, TXDBGrid, TXQRGrid

TXColumnReport.AutoHeight

Determines column which takes part in automatic row height adjustment.

```
property AutoHeight: Boolean;
```

Description
Set AutoHeight to True to cause the column takes part in automatic row height adjustment calculations when AutoRowHeight property is rhPartial.

Set AutoHeight to False when the column does'nt take part in automatic row height adjustment calculations when AutoRowHeight property is rhPartial.

See also: TXQRGrid.AutoRowHeight

TXColumnReport.Column

Specifies the TXColumn object that is associated with the column extension for a report.

```
property Column: TXColumn;
```

Description
Use Column to determine the column associated with the column report.

See also: TXColumn.Report

TXColumnReport.DialogOptions

Specifies additional column's options for TXDBPrintColumnsDialog component.

```
type
  TDialogOption = (doShowColumn, doItemEnabled);
  TDialogOptions = set of TDialogOption;
property DialogOptions: TDialogOptions;
```

Description
Set DialogOptions to include additional options for TXDBPrintColumnsDialog component. These are the possible values of DialogOptions:

Value	Meaning
doShowColumn	The column appears on TXDBPrintColumnsDialog's list.
doItemEnabled	TXDBPrintColumnsDialog's list item is enabled and Visible property can be changed in dialog.

See also: TXDBPrintColumnsDialog, TXColumn.DialogOptions

TXColumnReport.TotalMask

Specifies mask for column's total printed in the report.

```
property TotalMask: string;
```

Description
Use TotalMask property to format total value printed in the report. If no mask is specified the default formatting for the data type will be used. For more information on formatting numeric field please see FormatFloat function.

See also: TXColumnReport.TotalText, TXColumnReport.TotalType

TXColumnReport.TotalText

Specifies text or expression printed in column's total in the report.

```
property TotalText: string;
```

Description
Use TotalText property to specify text or QuickReport's expression printed in column's total depend on TotalType property setting.

See also: TXColumnReport.TotalMask, TXColumnReport.TotalType

TXColumnReport.TotalType

Specifies type of column's total printed in the report.

```
type
  TTotalType = (ttNone, ttSum, ttCount, ttMax, ttMin, ttAverage, ttText,
    ttExpression, ttGrowExpression);
property TotalType: TTotalType;
```

Description
Set the TotalType property to determine type of column's total printed in the report. These are the possible values of TotalType:

Value	Meaning
ttNone	None total is printed for the column.
ttSum	The sum of field's values is printed for the column.
ttCount	The count of records is printed for the column.
ttMax	The maximum field's value is printed for the column.
ttMin	The minimum field's value is printed for the column.
ttAverage	The avarage field's value is printed for the column.
ttText	The text specified in TotalText property is printed for the column.
ttExpression	The QuickReport's expression specified in TotalText is printed for the column. Value of expression is ResetAfterPrint.
ttGrowExpression	The QuickReport's expression specified in TotalText is printed for the column. Value of expression is NOT ResetAfterPrint.

See also: TXColumnReport.TotalMask, TXColumnReport.TotalText

TXColumnReport.Visible

Specifies whether the column may be visible in the report.

```
property Visible: Boolean;
```

Description
To hide a column in the report, set Visible to False. If Visible is True, and the column's Visible property is True, the column will appear in the report. This property only determines whether the column is allowed to be printed.

See also: TXColumnReport.IsPrinted

TXColumnReport.Create

Creates and initializes a column extension for a report.

```
constructor Create(Column: TXColumn);
```

Description
Call Create to instantiate an instance of TXColumnReport. Create takes a TXColumn instance as its argument.

Most applications need not create column extesion instances as these are instantiated by the associated column.

See also: TColumn

TXColumnReport.IsPrinted

Indicates whether the column is printed in the report.

```
function IsPrinted: Boolean;
```

Description
IsPrinted returns True if the column is visibled in the grid and Visible property is True.

See also: TXColumnReport.Visible

TXColumnReport.RestoreDefaults

Restores the column extesion's default settings.

```
procedure RestoreDefaults; virtual;
```

Description
RestoreDefaults reinitializes all column extesion's properties.

See also: TColumn.RestoreDefaults

TXDBGridFilter

TXDBGridFilter contains internal filter settings for the grid.

Unit
XDBGrids

Description
TXDBGrid has an associated TXDBGridFilter that holds additional parameters for internal filter settings.

See also: TXCustomDBGrid.FilterGrid

TXDBGridFilter.Active

Specifies whether or not the internal filter is active.

```
property Active: Boolean; {* ver. 6.0 *}
```

Description
Use Active to determine whether the internal filter is active. When Active property is True the filtering of DataSet is controlled by the grid. You should not modify Filter, Filtered and OnFilterRecord property directly in the DataSet linked to the grid when Active property is True, but you can modify Filter, Filtered and OnFilterRecord property of TXDBGrid instead.

The property Active is True when any FilterList is not empty or any FilterText contains filter condition and AutoUpdate property is True.

See also: TXColumn.FilterList, TXColumn.FilterText, TXDBGridFilter.AutoUpdate

TXDBGridFilter.AutoFilter

Specifies whether or not each column has auto-filter drop-down list or auto-filter form.

```
property AutoFilter: Boolean; {* ver. 6.0 *} {* ver. 7.1 *}
```

Description
Set AutoFilter on to show auto-filter drop-down button in title area of each column. When drop-down button is visible in title area the auto-filter list or auto-filter form is available for the column. Auto-filter list allows to check/uncheck values for the column to show/hide selected records in DataSet linked to the grid. Values checked in drop-down auto-filter list are stored in FilterList property. Auto-filter form allows to build a filter condition. The condition selected in auto-filter form is stored in FilterText property.

When AutoFilter is False you can populate filter list for selected column with ExpandStyle = cesFilterList and you can populate filter form for the column with ExpandStyle = cesFilterForm.

See also: TXDBGridFilter.Filtered, TXDBGridFilter.AutoFilters, TXColumn.ExpandStyle, TXColumn.FilterList, TXColumn.FilterText, TXColumnTitle.AutoFilter

TXDBGridFilter.AutoFilters

Specifies the available filter styles for the grid.

```
type
  TFilterStyle = (fsFilterList, fsFilterForm); {* ver. 7.1 *}
  TFilterStyles = set of TFilterStyle;
property AutoFilters: TFilterStyles; {* ver. 7.1 *}
```

Description
Set AutoFilters to include available filter styles for the grid. AutoFilters is a set drawn from the following values:

Value	Meaning
fsFilterList	Include fsFilterList to allow using filter list to check/uncheck values for the column.
fsFilterForm	Include fsFilterForm to allow using universal filter form to build a filter condition.

See also: TXDBGridFilter.AutoFilter, TXDBGridFilter.Options

TXDBGridFilter.AutoUpdate

Specifies whether or not internal grid filter system is handled by the grid.

```
property AutoUpdate: Boolean; {* ver. 6.0 *}
```

Description
When AutoUpdate is True, the internal grid filter system is automatically handled by the grid.

When AutoUpdate is False, each change in FilterList or FilterText property is notified by OnFilterChanged event and the change should be updated in DataSet in your event handler (in any way). After all you should notify the grid by calling FilterUpdated that the filter change was updated in DataSet.

See also: TXDBGridFilter.Filtered, TXCustomDBGrid.OnFilterChanged

TXDBGridFilter.Clauses

Specifies the acceptable conditions in the auto-filter form.

```
type
  TXDBGridFilterClause = (fcNoFilter, fcContains, fcDoesNotContain,
    fcStartsFrom, fcDoesNotStartFrom, fcEndsWith, fcDoesNotEndWith, fcIsEqual,
    fcIsNotEqual, fcIsGreaterThen, fcIsGreaterThenOrEqual, fcIsLessThen,
    fcIsLessThenOrEqual, fcIsOnTheList, fcIsNotOnTheList, fcIsEmpty,
    fcIsNotEmpty);
  TXDBGridFilterClauses = set of TXDBGridFilterClause;
property Clauses: TXDBGridFilterClauses; {* ver. 7.1 *}
```

Description
Set Clauses to include the acceptable conditions in the auto-filter form. Clauses is a set drawn from the following values:

Value	Meaning
fcNoFilter	No filter.
fcContains	Contains (text).
fcDoesNotContain	Does not contain (text).
fcStartsFrom	Starts from (text).
fcDoesNotStartFrom	Does not starts form (text).
fcEndsWith	Ends with (text).
fcDoesNotEndWith	Does not ends with (text).
fcIsEqual	Is equal (value).
fcIsNotEqual	Is not equal (value).
fcIsGreaterThen	Is greater then (value).
fcIsGreaterThenOrEqual	Is greater then or equal (value).
fcIsLessThen	Is less then (value).
fcIsLessThenOrEqual	Is less then or equal (value).
fcIsOnTheList	Is on the list (list of values).
fcIsNotOnTheList	Is not on the list (list of values).
fcIsEmpty	Is empty (no value).
fcIsNotEmpty	Is not empty (no value).

See also: TXDBGridFilter.AutoFilters, TXDBGridFilter.Periods

TXDBGridFilter.Columns

Specifies grid columns with active FilterList or FilterText.

```
property Columns: TList; {* ver. 6.0 *}
```

Description
Use Columns property to examine grid's columns when FilterList is not empty or FilterText has defined filter condition.

See also: TXColumn.FilterList, TXColumn.FilterText

TXDBGridFilter.DecimalSeparator

Specifies decimal separator in the filter clause.

```
property DecimalSeparator: Boolean; {* ver. 7.1 *}
```

Description
When DecimalSeparator is True the filtered data set requires using of system DecimalSeparator from SysUtils module otherwise the dot sign is used as the decimal separator in the filter clause.

See also: TXDBGridFilter.Filtered

TXDBGridFilter.DropDownRows

Specifies the number of rows displayed in the filter's drop-down list.

```
property DropDownRows: Integer; {* ver. 6.0 *}
```

Description
DropDownRows determines the number of rows displayed in the filter's drop-down list.

When DropDownRows property is 0 (default) and drop-down list has more then MaxDropDownRows items, then MaxDropDownRows items is displayed in the filter's drop-down list, otherwise all list items are displayed.

See also: TXDBGridFilter.DropDownWidth, TXDBGridFilter.Options

TXDBGridFilter.DropDownWidth

Specifies a width of the filter's drop-down list.

```
property DropDownWidth: Integer; {* ver. 6.0 *}
```

Description
DropDownWidth determines a width of the filter's drop-down list.

When DropDownWidth property is 0 (default) and auto-calculated width of the filter's drop-down list is greater then MaxDropDownWidth, then displayed width of the filter's drop-down list is limited to MaxDropDownWidth.

See also: TXDBGridFilter.DropDownRows, TXDBGridFilter.Options

TXDBGridFilter.Filtered

Specifies whether or not the internal grid filter system is active.

```
property Filtered: Boolean; {* ver. 6.0 *}
```

Description
Set Filtered to False to disable internal grid filter system.

See also: TXDBGridFilter.Active, TXDBGridFilter.AutoUpdate, TXCustomDBGrid.Filtered

TXDBGridFilter.FilterText

Specifies filter condition for all filtered columns in the grid.

```
property FilterText: string; {* ver. 6.0 *}
```

Description
The FilterText property specifies conjunction of TXColumn.FilterText property for all filtered columns in the grid.

See also: TXDBGridFilter.FullFilter, TXColumn.FilterText

TXDBGridFilter.FullFilter

Specifies full filter condition for internal grid's filter and external DataSet filter.

```
property FullFilter: string; {* ver. 6.0 *}
```

Description
The FullFilter property specifies conjunction of FilterText property and Filter property, if Filtered property is True.

See also: TXDBGridFilter.FilterText, TXCustomDBGrid.Filter

TXDBGridFilter.Grid

Indicates the grid that contains this object.

```
property Grid: TXCustomDBGrid; {* ver. 6.0 *}
```

Description
Use the Grid property to access the grid that contains this object.

See also: TXCustomDBGrid.FilterGrid

TXDBGridFilter.Options

Specifies various properties of the filter drop-down list.

```
type
  TXDBGridFilterOption = (foAutoApplyList, foListAnimation, foListAlignment,
    foOverlayedList, foLoadAllItems, foCheckedFirst, foAddSelected,
    foIncrementalList); {* ver. 7.1 *} {* ver. 7.8 *}
  TXDBGridFilterOptions = set of TXDBGridFilterOption;
property Options: TXDBGridFilterOptions; {* ver. 6.0 *}
```

Description
Set Options to include the desired properties for the filter drop-down list. Options is a set drawn from the following values:

Value	Meaning
foAutoApplyList	Include foAutoApplyList to automatically apply changes (checked/unchecked values) in DataSet linked to the grid. Exclude foAutoApplyList to show OK and Cancel button in the filter list.
foListAnimation	Include foListAnimation to animate drop-down process of the filter list.
foListAlignment	Include foListAlignment to align width of the filter list with the width of the column, if possible.
foOverlayedList	Include foOverlayedList to drop-down filter list over the column.
foLoadAllItems	Include foLoadAllItems to show on filter list all possible values for the column even they are excluded by current filter settings in other columns. You can also show/hide excluded values when filter list is visible by click on the left upper checkbox.
foCheckedFirst	Include foCheckedFirst to load checked items on the top of the filter list when the list is drop down.
foAddSelected	Include foAddSelected to add over the filer list a special position "Add selected". The special position "Add selected" allows to make checked on the

	list all values that are multiselected in this column in the grid. You can see this special position over the list only when the SelectedRows list is not empty.
foIncrementalList	Include foIncrementalList to activate Incremental Search function in all filter list in the grid. See also dgIncrementalList option in Options property and loIncrementalList in ListOptions. {* ver. 7.8 *}

See also: TXDBGridFilter.DropDownRows, TXDBGridFilter.DropDownWidth

TXDBGridFilter.Periods

Specifies the acceptable periods in the auto-filter form.

```
type
  TXDBGridFilterPeriod = (fpNoPeriod, fpMinute, fpHour, fpDay, fpWeek,
    fpMonth, fpQuarter, fpSemester, fpYear, fpDecade, fpOther, fp10Mins,
    fp15Mins, fp20Mins, fp30Mins, fpHours, fpDays, fpWeeks, fpMonths,
    fpQuarters, fpSemesters, fpYears, fpDecades, fpOthers);
  TXDBGridFilterPeriods = set of TXDBGridFilterPeriod;
property Periods: TXDBGridFilterPeriods; {* ver. 7.1 *}
```

Description
Set Periods to include the acceptable periods in the auto-filter form. Periods is a set drawn from the following values:

Value	Meaning
fpNoPeriod	No period.
fpMinute	Group of minutes.
fpHour	Group of hours.
fpDay	Group of days.
fpWeek	Group of weeks.
fpMonth	Group of months.
fpQuarter	Group of quarters.
fpSemester	Group of semesters.
fpYear	Group of years.
fpDecade	Group of decades.
fpOther	Group of other periods.
fp10Mins	Periods of 10 minutes.
fp15Mins	Periods of 15 minutes.
fp20Mins	Periods of 20 minutes.
fp30Mins	Periods of 30 minutes.
fpHours	Periods of hours.
fpDays	Periods of the day.
fpWeeks	Periods of the week.
fpMonths	Periods of the month.
fpQuarters	Periods of the quarter.
fpSemesters	Periods of the semester.
fpYears	Periods of the year.
fpDecades	Periods of the decade.

fpOthers Additional periods.

See also: TXDBGridFilter.AutoFilters, TXDBGridFilter.Clauses

TXDBGridFilter.Settings

Contains lines of text for filter layout settings.

```
property Settings: TStrings; {* ver. 8.0 *}
```

Description
Settings is a TStrings object designed for store and retrieve filter layout settings as name-value pairs. For more information on name-value pairs, refer to Values property.

The Settings object contains several lines with filter settings (Column.FieldName, Column.FilterText, Column.FilterList) for each filtered column. Name of each line determine one filtered column in the XDBGrid (e.g. "Filter1", "Filter2", ...). The line with name "Filters" contains number of columns in the grid with active filter (Column.Filtered = True).

This convention corresponds to the format of configuration files (*.ini). For example:

Filters=2
Filter1=Country,Country LIKE '%US%',
Filter2=State,,AL,CA,FL,OR,TX

The Settings object is designed to store and retrieve filter layout to/from ini file or to/from registry but you can also use TStrings methods like: LoadFromFile/SaveToFile, LoadFromStream/SaveToStream, GetTextStr/SetTextStr (or Text property) to load/save filter Settings to text file, any stream (BLOB field) or memory variable. Each time you are reading Settings property the current filter settings are retrieved from XDBGrid's properties. Each time you are setting new Settings property the XDBGrid's columns properties are modified. After you load Settings.Text property from the text file, stream or variable you need to call ApplySettings method to apply changes to the grid columns.

The contains of Settings property may be a part of Settings.Layout property.

See also: TXCustomDBGrid.FilterGrid, TXCustomDBGrid.Columns, TXDBGridFilter.ApplySettings

TXDBGridFilter.Visible

Specifies whether or not the grid filter icons are visible.

```
property Visible: Boolean; {* ver. 8.0 *}
```

Description
The Visible property specifies whether grid filter icons are visible. The Visible property returns True when AutoFilter property is True and Filtered property is True.

The Visible is a read-only property.

See also: TXDBGridFilter.Active, TXDBGridFilter.AutoActive, TXCustomDBGrid.Filtered

TXDBGridFilter.Create

Creates and initializes a FilterGrid object.

```
constructor Create(AGrid: TXCustomDBGrid); {* ver. 6.0 *}
```

Description
Call Create to instantiate an instance of TXDBGridFilter. Create takes a TXCustomDBGrid instance as its argument. You not need to create FilterGrid directly. This is instantiated by XDBGrid.

See also: TXCustomDBGrid, TXDBGridFilter.Destroy

TXDBGridFilter.Destroy

Destroys an instance of TXDBGridFilter.

```
destructor Destroy; {* ver. 6.0 *}
```

Description
Destroy frees the helper objects of the TXDBGridFilter before destroying the instance. The user never need to call this method directly.

See also: TXDBGridFilter.Create

TXDBGridFilter.ApplySettings

Applies changes in filter layout stored in Settings buffer.

```
procedure ApplySettings; {* ver. 8.0 *}
```

Description
Use ApplySettings to apply new filters Settings in the grid. This procedure is useful after you load new filter layout to Settings buffer by calling LoadFromFile, LoadFromStream methods or after you setting the new value for property Text.

See also: TXDBGridFilter.AutoFilter, TXColumn.FilterList

TXDBGridFilter.ClearAutoFilter

Clears active FilterList for each column in the grid.

```
procedure ClearAutoFilter(Part: TFilterStyles = [fsFilterList]){* ver. 7.3 *};
{* ver. 6.8 *}
```

Description
Use ClearAutoFilter method to clear FilterList and/or FilterForm for each column in the grid.

See also: TXDBGridFilter.AutoFilter, TXColumn.FilterList

TXDBGridFilter.JoinFilter

Joins FilterText with other filter string.

```
function JoinFilter(Filtered: Boolean; const Filter: string): string; {* ver. 6.0 *}
```

Description
Use JoinFilter to return conjunction of current FilterText property with other Filter string when Filtered parameter is True. JoinFilter is used to return FullFilter condition.

See also: TXDBGridFilter.FilterText, TXDBGridFilter.FullFilter

TXDBGridFilter.SwitchAutoFilters

Include or exclude AutoFilters depending on State.

```
procedure SwitchAutoFilters(Part: TFilterStyles; State: Boolean); {* ver. 7.1 *}
```

Description
Use SwitchAutoFilters to include or exclude the Part of AutoFilters property. When State is True the Part options are included, otherwise excluded.

See also: TXDBGridFilter.AutoFilters

TXDBGridFilter.SwitchClauses

Include or exclude Clauses depending on State.

```
procedure SwitchClauses(Part: TXDBGridFilterClauses; State: Boolean); {* ver. 7.1 *}
```

Description
Use SwitchClauses to include or exclude the Part of Clauses property. When State is True the Part options are included, otherwise excluded.

See also: TXDBGridFilter.Clauses

TXDBGridFilter.SwitchOptions

Include or exclude Options depending on State.

```
procedure SwitchOptions(Part: TXDBGridFilterOptions; State: Boolean); {* ver. 6.0 *}
```

Description
Use SwitchOptions to include or exclude the Part of Options property. When State is True the Part options are included, otherwise excluded.

See also: TXDBGridFilter.Options

TXDBGridFilter.SwitchPeriods

Include or exclude Periods depending on State.

```
procedure SwitchPeriods(Part: TXDBGridFilterPeriods; State: Boolean); {* ver. 7.1 *}
```

Description

Use SwitchPeriods to include or exclude the Part of Periods property. When State is True the Part options are included, otherwise excluded.

_____

See also: TXDBGridFilter.Periods

TXDBGridSearch

TXDBGridSearch represents a search settings for the grid.

Unit
XDBGrids

Description
TXDBGrid has an associated TXDBGridSearch object that holds settings for the search panel.

See also: TXCustomDBGrid.Search

TXDBGridSearch.Active

Specifies whether or not the search system is active.

```
property Active: Boolean; {* ver. 6.2 *}
```

Description
Use Active to determine whether the search system is active. When Active property is True the search Text can be visible in the grid and the search panel can be shown in the grid by the user.

See also: TXDBGridSearch.Text, TXDBGridSearch.Visible

TXDBGridSearch.Color

Specifies the color of the search text.

```
property Color: TColor; {* ver. 6.2 *}
```

Description
Set Color to specify the color of the search text in the grid.

The Color property can handle two special colors: clDefault, clNone.
Select clDefault value to use DefaultColor value.
Select clNone value when the search text should not be highlighted.

See also: TXDBGridSearch.Text, TXDBGridSearch.DefaultColor

TXDBGridSearch.FilterActive

Specifies whether or not the search filter system is active.

```
property FilterActive: Boolean; {* ver. 6.5 *}
```

Description
Use FilterActive to determine whether the search filter system is active. When FilterActive property is True only rows that contain search Text are visible in the grid - other rows are hidden.

See also: TXDBGridSearch.Active, TXDBGridSearch.Visible

TXDBGridSearch.Grid

Indicates the grid that contains this object.

```
property Grid: TXCustomDBGrid; {* ver. 6.2 *}
```

Description
Use the Grid property to access the grid that contains this object.

See also: TXCustomDBGrid.Search

TXDBGridSearch.Incremental

Specifies the incremental search is active in the grid.

```
property Incremental: Boolean; {* ver. 7.1 *}
```

Description
Use Incremental to determine the incremental search is active in the grid. Set Incremental to True to search text after each key pressed. Set Incremental to False to search text after Enter key pressed in Search Panel.

Hint. Property Incremental works also when soFilterRows and soCurrentCol are included in Search.Options.

Starting from version 7.8 property Incremental can be used to incremental search in the grid even when Search Panel is not Visible. To activate incremental search when Search Panel is not visible soFilterRows option must be included in Search.Options and dgEditing option must be excluded in Grid.Options alternatively dgShowEditorByCharOff must be included in Grid.OptionsEx. {* ver. 7.8 *}

See also: TXDBGridSearch.Options

TXDBGridSearch.List

Holds list of textes searched by the user.

```
property List: TStrings; {* ver. 6.2 *}
```

Description
Use List property to read or change textes searched by the user in search panel.

See also: TXDBGridSearch.Text

TXDBGridSearch.ListWidth

Specifies the default width of the search list.

```
property ListWidth: Integer; {* ver. 7.0 *}
```

Description
Use ListWidth property to change the default width of the search list visible on the search panel.

See also: TXDBGridSearch.List

TXDBGridSearch.Options

Specifies various properties of the seaching in the grid.

```
type
  TXSearchOption = (soCaseSensitive, soWholeWords, soWrapAround,
    soSkipMessage, soCurrentCol, soFilterRows); {* ver. 6.5 *}{* ver. 7.1 *}
  TXSearchOptions = set of TXSearchOption;
property Options: TXSearchOptions; {* ver. 6.2 *}
```

Description
Set Options to include the desired properties for the searching in the grid. The possible values of Options are:

Value	Meaning
soCaseSensitive	Include soCaveSensitive to search text with differentiating between capital and lowercase letters.
soWholeWords	Include soWholeWords to search text occurences only for whole words.
soWrapAround	Include soWrapAround to search next text occurence from the beginning after the last.
soSkipMessage	Include soSkipMessage to skip message when the searched text has not been found.
soCurrentCol	Include soCurrentCol to search/filter text occurences in the current column only.
soFilterRows	Include soFilterRows to filter rows that contain search text.

See also: TXDBGridSearch.Text, TXDBGridSearch.Incremental

TXDBGridSearch.PanelColor

Specifies the background color of the search panel.

```
property PanelColor: TColor; {* ver. 7.0 *}
```

Description
Set PanelColor to specify the background color of the search panel in the grid.

The PanelColor property can handle one special color: clDefault.
Select clDefault value to use DefaultPanelColor value.

See also: TXDBGridSearch.Color, TXDBGridSearch.DefaultPanelColor

TXDBGridSearch.PanelOnTop

Specifies the search panel is visible on the top of the grid.

```
property PanelOnTop: Boolean; {* ver. 7.0 *}
```

Description
Use PanelOnTop to determine the search panel is visible on the top of the grid. When property PanelOnTop is False (default) the search panel in visible on the bottom of the grid. In design-time the search panel is always hidden. Notice: When Custom Style is active property PanelOnTop is ignored.

See also: TXCustomDBGrid.ShowSearchPanel, TXDBGridSearch.PanelVisible

TXDBGridSearch.PanelVisible

Specifies the search panel is visible in the grid.

```
property PanelVisible: Boolean; {* ver. 6.8 *}
```

Description
Use PanelVisible to determine the search panel is visible in the grid. Set PanelVisible to True to show search panel in run-time. In design-time the search panel is always hidden.

See also: TXCustomDBGrid.ShowSearchPanel, TXDBGridSearch.Visible, TXDBGridSearch.PanelOnTop

TXDBGridSearch.Text

Specifies text to search in the grid.

```
property Text: string; {* ver. 6.2 *}
```

Description
The Text property specifies text to search in the grid. Each occurence of text in the grid is highlighted in Color. Only first occurence in the cell is highlighted. The specified Text is searching with using soCaseSensitive and soWholeWords from Options property.

See also: TXDBGridSearch.Color, TXDBGridSearch.Options

TXDBGridSearch.Visible

Specifies the search text is visible in the grid.

```
property Visible: Boolean; {* ver. 6.2 *}
```

Description
Use Visible to determine the search text is visible in the grid. The Visible is True when Active or Incremental {* ver. 7.8 *} property is True and Text property is not empty.

The Visible is a read-only property.

See also: TXDBGridSearch.Active, TXDBGridSearch.Text

TXDBGridSearch.VisibleOptions

Specifies various visible options of the seaching in the grid.

```
property VisibleOptions: TXSearchOptions; {* ver. 7.5 *}
```

Description
Set VisibleOptions to include options for the searching in the grid which are visible on SearchPanel. The possible values are corresponding to Options property.

See also: TXDBGridSearch.Options

TXDBGridSearch.Create

Creates and initializes a Search object.

```
constructor Create(AGrid: TXCustomDBGrid); {* ver. 6.2 *}
```

Description
Call Create to instantiate an instance of TXDBGridSearch. Create takes a TXCustomDBGrid instance as its argument. You not need to create Search directly. This is instantiated by XDBGrid.

See also: TXCustomDBGrid, TXDBGridSearch.Destroy

TXDBGridSearch.Destroy

Destroys an instance of TXDBGridSearch.

```
destructor Destroy; {* ver. 6.2 *}
```

Description
Destroy frees the helper objects of the TXDBGridSearch before destroying the instance. The user never need to call this method directly.

See also: TXDBGridSearch.Create

TXDBGridSearch.DefaultColor

Indicates default color for highlighted search text.

```
function DefaultColor: TColor; {* ver. 6.2 *}
```

Description
Call DefaultColor to get default color of highlighted search text. DefaultColor is used when Color property is set to clDefault color.

See also: TXDBGridSearch.Color

TXDBGridSearch.DefaultPanelColor

Indicates default background color for the search panel.

```
function DefaultPanelColor: TColor; {* ver. 7.0 *}
```

Description
Call DefaultPanelColor to get default background color of the search panel. DefaultPanelColor is used when PanelColor property is set to clDefault color.

See also: TXDBGridSearch.PanelColor

TXDBGridSearch.SwitchOptions

Include or exclude Options depending on State.

```
procedure SwitchOptions(Part: TXSearchOptions; State: Boolean); {* ver. 6.2 *}
```

Description
Use SwitchOptions to include or exclude the Part of Options property. When State is True the Part options are included, otherwise excluded.

See also: TXDBGridSearch.Options

TXDBGridSearch.SwitchVisibleOptions

Include or exclude VisibleOptions depending on State.

```
procedure SwitchVisibleOptions(Part: TXSearchOptions; State: Boolean); {* ver. 7.5 *}
```

Description
Use SwitchVisibleOptions to include or exclude the Part of VisibleOptions property. When State is True the Part options are included, otherwise excluded.

See also: TXDBGridSearch.VisibleOptions

TXDBGridTreeView

TXDBGridTreeView represents a tree view in the grid.

Unit
XDBGrids

Description
TXDBGrid has an associated TXDBGridTreeView that holds additional parameters for tree view in the grid.

See also: TXCustomDBGrid.TreeView

TXDBGridTreeView.Active

Specifies whether or not the tree view is active in the grid.

```
property Active: Boolean; {* ver. 7.0 *}
```

Description
Use Active to determine whether the tree view is active in the grid. Active is True when KeyFields and ParentFields has been determined, AutoActive is True and DataSet linked to the grid is open.

The Active is a read-only property.

See also: TXDBGridTreeView.AutoActive

TXDBGridTreeView.AutoActive

Specifies whether the tree view can be active in the grid.

```
property AutoActive: Boolean; {* ver. 7.0 *}
```

Description
Use AutoActive to activate/deactivate the tree view in the grid. The tree view is active in the grid when KeyFields and ParentFields has been determined, AutoActive is True and DataSet linked to the grid is open. Set AutoActive to False to deactivate tree view without changes other settings.

See also: TXDBGridTreeView.Active

TXDBGridTreeView.CellExpand

Specifies whether the TreeViiew column allows to expand/collapse rows by clicking on the whole expandable cell.

```
property CellExpand: Boolean; {* ver. 7.1 *}
```

Description
Set CellExpand to True to allow clicking on whole cell to expand/collapse TreeView rows. Set CellExpand to False to expand/collapse TreeView rows by clicking on expand box only. {* ver. 7.1 *}

See also: TXDBGridTreeView.IsRowExpandable, TXColumn.ListOptions

TXDBGridTreeView.AutoUpdate

Specifies whether the tree view is automaticaly updated.

```
property AutoUpdate: Boolean; {* ver. 7.0 *}
```

Description
Set AutoUpdate to True to automaticaly update the tree view in the grid. When AutoUpdate is False, you need to call UpdateTreeView method when key data changed in the grid.

See also: TXCustomDBGrid.UpdateTreeView, TXCustomDBGrid.KeyFields, TXDBGridTreeView.ParentFields

TXDBGridTreeView.ColumnNames

Indicate field(s) for tree view in dataset linked to the grid.

```
property ColumnNames: string; {* ver. 7.0 *}
property ColumnNames: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
ColumnNames indicate field(s) for the tree view in dataset linked to the grid. ColumnNames may contain more than one tree view field separated by semicolon. When ColumnNames property is defined the ExpandBoxTree property is automaticaly controlled for selected grid's columns.

See also: TXDBGridTreeView.ParentFields, TXColumn.ExpandBoxTree

TXDBGridTreeView.Grid

Indicates the grid that contains this object.

```
property Grid: TXCustomDBGrid; {* ver. 7.0 *}
```

Description
Use the Grid property to access the grid that contains this object.

See also: TXCustomDBGrid.TreeView

TXDBGridTreeView.KeyFields

Indicates the key field or fields in dataset linked to the grid.

```
property KeyFields: string; {* ver. 8.0 *}
property KeyFields: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
KeyFields specifies the field or fields which allows to identify any record in dataset linked to the grid. KeyFields in TXDBGridTreeView class duplicates KeyFields from TXCustomDBGrid class just for convenience. The fields specified in ParentFields must be of the same types as KeyFields, or the tree view can't work.

See also: TXCustomDBGrid.KeyFields, TXDBGridTreeView.ParentFields, TXDBGridTreeView.ColumnNames

TXDBGridTreeView.Level

Specifies the depth level in the tree view for the current row in the grid.

```
property Level: Integer; {* ver. 7.0 *}
```

Description
Use Level to determine depth level in the tree view for the current row in the grid. The main level in the tre view has number 0.

The Level is a read-only property.

See also: TXDBGridTreeView.MaxLevel, TXDBGridTreeView.OpenToLevel

TXDBGridTreeView.MaxLevel

Specifies maximum depth level in the tree view for all rows in the grid.

```
property MaxLevel: Integer; {* ver. 7.0 *}
```

Description
Use MaxLevel to determine maximum depth level in the tree view for all rows in the grid.

The MaxLevel is a read-only property.

See also: TXDBGridTreeView.Level, TXDBGridTreeView.OpenToLevel

TXDBGridTreeView.OpenToLevel

Specifies depth level in the tree view visible when data set is opened.

```
property OpenToLevel: Integer; {* ver. 7.0 *}
```

Description
Use OpenToLevel to determine depth level in the tree view visible when data set is opened. Set OpenToLevel = -1 (default) to full expand tree view. Set OpenToLevel = 0 to collapse tree view. Set OpenToLevel = N to expand only N levels in the tree view. OpenToLevel value is applyed always when data set is opened.

See also: TXDBGridTreeView.Level, TXDBGridTreeView.MaxLevel, TXDBGridTreeView.ShowToLevel

TXDBGridTreeView.ParentFields

Indicates the parent field for tree view in dataset linked to the grid.

```
property ParentFields: string; {* ver. 7.0 *}
property ParentFields: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
ParentFields contain key values of parent row for tree view in the grid. ParentFields are fields in DataSet linked to the grid which values must match KeyFields. The fields specified in ParentFields must be of the same types as KeyFields, or the tree view can't work.

See also: TXCustomDBGrid.KeyFields, TXDBGridTreeView.ColumnNames

TXDBGridTreeView.RowLevelFieldName

Specifies field name for row level field in tree view.

```
property RowLevelFieldName: string; {* ver. 7.0 *}
```

Description
Set RowLevelFieldName to change default field name for row level field in tree view. You not need to change this name when it's not necessary.

Notice. To support tree view functionality TXDBGrid must add to data set 3 invisible InternalCalc fields in run-time.
The tree view functionality works only when data set fully support InternalCalc fields including sorting on these fields.

See also: , TXDBGridTreeView.BeforeOpen, TXDBGridTreeView.RowOrderFieldName, TXDBGridTreeView.RowStateFieldName

TXDBGridTreeView.RowOrderFieldName

Specifies field name for row order field in tree view.

```
property RowOrderFieldName: string; {* ver. 7.0 *}
```

Description
Set RowOrderFieldName to change default field name for row order field in tree view. You not need to change this name when it's not necessary.

Notice. To support tree view functionality TXDBGrid must add to data set 3 invisible InternalCalc fields in run-time.
The tree view functionality works only when data set fully support InternalCalc fields including sorting on these fields.

See also: TXDBGridTreeView.BeforeOpen, TXDBGridTreeView.RowLevelFieldName, TXDBGridTreeView.RowOrderFieldSize, TXDBGridTreeView.RowStateFieldName

TXDBGridTreeView.RowOrderFieldSize

Specifies field size for row order field in tree view.

```
property RowOrderFieldSize: Integer; {* ver. 7.0 *}
```

Description
Set RowOrderFieldSize to change default field size for row order field in tree view. You need to increase this size only when tree view has very many depth levels.

Notice. To support tree view functionality TXDBGrid must add to data set 3 invisible InternalCalc fields in run-time.
The tree view functionality works only when data set fully support InternalCalc fields including sorting on these fields.

See also: TXDBGridTreeView.RowOrderFieldName

TXDBGridTreeView.RowStateFieldName

Specifies field name for row state field in tree view.

```
property RowStateFieldName: string; {* ver. 7.0 *}
```

Description
Set RowStateFieldName to change default field name for row state field in tree view. You not need to change this name when it's not necessary.

Notice. To support tree view functionality TXDBGrid must add to data set 3 invisible InternalCalc fields in run-time.
The tree view functionality works only when data set fully support InternalCalc fields including sorting on these fields.

See also: TXDBGridTreeView.BeforeOpen, TXDBGridTreeView.RowLevelFieldName, TXDBGridTreeView.RowOrderFieldName

TXDBGridTreeView.Create

Creates and initializes a TreeView object.

```
constructor Create(AGrid: TXCustomDBGrid); {* ver. 7.0 *}
```

Description
Call Create to instantiate an instance of TXDBGridTreeView. Create takes a TXCustomDBGrid instance as its argument. You not need to create TreeView directly. This is instantiated by XDBGrid.

See also: TXCustomDBGrid, TXDBGridTreeView.Destroy

TXDBGridTreeView.Destroy

Destroys an instance of TXDBGridTreeView.

```
destructor Destroy; {* ver. 7.0 *}
```

Description
Destroy frees the helper objects of the TXDBGridTreeView before destroying the instance. The user never need to call this method directly.

See also: TXCustomDBGrid, TXDBGridTreeView.Create

TXDBGridTreeView.BeforeOpen

Add tree view fields before opening data set linked to the grid.

```
procedure BeforeOpen(DataSet: TDataSet); {* ver. 7.0 *}
```

Description
Use BeforeOpen to add tree view fields to data set linked to the grid before the data set is first time open. You not need to call this method unless data set can't be re-open by TXDBGrid. When you need to call this method you should write BeforeOpen event handler for data set linked to the grid:

```
procedure TForm1.ClientDataSet1BeforeOpen(DataSet: TDataSet);
begin
  XDBGrid1.TreeView.BeforeOpen(DataSet);
end;
```

Notice. To support tree view functionality TXDBGrid must add to data set 3 invisible InternalCalc fields in run-time.
The tree view functionality works only when data set fully support InternalCalc fields including sorting on these fields.

See also: TXDBGridTreeView.RowLevelFieldName, TXDBGridTreeView.RowOrderFieldName, TXDBGridTreeView.RowStateFieldName

TXDBGridTreeView.CollapseRow

Collapse current row in the tree view.

```
procedure CollapseRow(Recurse: Boolean); {* ver. 7.0 *}
```

Description
Use CollapseRow to collapse current row in the tree view. When Recurse parameter is False only one row is collapsed. When Recurse parameter is True each expandable rows on next depth levels are also collapsed. CollapseRow method works only when IsRowExpandable is True and IsRowExpanded is True.Use Shift+Click on collapse box to collapse current row recursively.

See also: TXDBGridTreeView.ExpandRow, TXDBGridTreeView.ToggleRow, TXDBGridTreeView.FullCollapse

TXDBGridTreeView.ExpandRow

Expand current row in the tree view.

```
procedure ExpandRow(Recurse: Boolean); {* ver. 7.0 *}
```

Description
Use ExpandRow to expand current row in the tree view. When Recurse parameter is False only one row is expanded. When Recurse parameter is True each expandable rows on next depth levels are also expanded. ExpandRow method works only when IsRowExpandable is True and IsRowExpanded is False. Use Shift+Click on expand box to expand current row recursively.

See also: TXDBGridTreeView.CollapseRow, TXDBGridTreeView.ToggleRow, TXDBGridTreeView.FullExpand

TXDBGridTreeView.FocusRow

Make current record focused in the TreeView.

```
procedure FocusRow; {* ver. 7.1 *}
```

Description
Call FocusRow to make current record focused in the TreeView. To make current record visible in the TreeView all parents are expanded to the first Level of tree. To make current row focused in the TreeView all other unnecessary rows will be collapsed. Use Ctrl+Click on expand/collapse box to make current row focused.

See also: TXDBGridTreeView.ShowRecord, TXDBGridTreeView.FullCollapse, TXDBGridTreeView.FullExpand

TXDBGridTreeView.FullCollapse

Collapse tree view to minimum depth level.

```
procedure FullCollapse; {* ver. 7.0 *}
```

Description
Use FullCollapse to collapse tree view to minimum depth level. FullCollapse method is equivalent ShowToLevel(0).

See also: TXDBGridTreeView.FullExpand, TXDBGridTreeView.ShowToLevel, TXDBGridTreeView.CollapseRow

TXDBGridTreeView.FullExpand

Expand tree view to maximum depth level.

```
procedure FullExpand; {* ver. 7.0 *}
```

Description
Use FullExpand to expand tree view to maximum depth level. FullExpand method is equivalent ShowToLevel(-1).

See also: TXDBGridTreeView.FullCollapse, TXDBGridTreeView.ShowToLevel, TXDBGridTreeView.ExpandRow

TXDBGridTreeView.IsRowExpanded

Determine expanded state of current row in the tree view.

```
function IsRowExpanded: Boolean; {* ver. 7.0 *}
```

Description
Use IsRowExpanded to determine is the current row in the tree view expanded or collapsed. IsRowExpanded return True when the current row has visible children.

See also: TXDBGridTreeView.IsRowExpandable, TXDBGridTreeView.IsRowVisible

TXDBGridTreeView.IsRowExpandable

Determine expandable state of current row in the tree view.

```
function IsRowExpandable: Boolean; {* ver. 7.0 *}
```

Description
Use IsRowExpandable to determine is the current row in the tree view can be expanded/collapsed. IsRowExpandable return True when the current row has children.

See also: TXDBGridTreeView.IsRowExpanded, TXDBGridTreeView.IsRowVisible

TXDBGridTreeView.IsRowVisible

Determine visible state of current row in the tree view.

```
function IsRowVisible: Boolean; {* ver. 7.0 *}
```

Description
Use IsRowVisible to determine is the current row in the tree view visible or not. IsRowVisible return True when parent row is visible and expanded.

See also: TXDBGridTreeView.IsRowExpandable, TXDBGridTreeView.IsRowExpanded

TXDBGridTreeView.ShowRecord

Make record visible in the TreeView.

```
procedure ShowRecord(const KeyValues: Variant; FocusRow: Boolean = False); {*
ver. 7.1 *}
```

Description
Call ShowRecord to make record visible in the TreeView. Specify parameter KeyValues to identify record in DataSet linked to the grid. To make record visible in the TreeView all parents are expanded to the first Level of tree. When optional parameter FocusRow is True all other unnecessary rows will be collapsed.

See also: TXDBGridTreeView.FocusRow, TXDBGridTreeView.FullCollapse, TXDBGridTreeView.FullExpand

TXDBGridTreeView.ShowToLevel

Expand/Collapse tree view to depth level specified in parameter.

```
procedure ShowToLevel(Level: Integer); {* ver. 7.0 *}
```

Description
Call ShowToLevel to expand/collapse tree view to depth level specified in Level parameter. Set Level = -1 to full expand tree view. Set Level = 0 to collapse tree view. Set Level = N to expand only N levels in the tree view.

See also: TXDBGridTreeView.OpenToLevel, TXDBGridTreeView.FullCollapse, TXDBGridTreeView.FullExpand

TXDBGridTreeView.ToggleRow

Expand or collapse current row in the tree view.

```
procedure ToggleRow(Recurse: Boolean); {* ver. 7.0 *}
```

Description
Use ToggleRow to expand current row in the tree view when the current row is now collapsed or to collapse current row in the tree view when the current row is now expanded. When Recurse parameter is False only one row is expanded/collapsed. When Recurse parameter is True each expandable rows on next depth levels are also expanded/collapsed. ToggleRow method works only when IsRowExpandable is True.

See also: TXDBGridTreeView.ExpandRow, TXDBGridTreeView.CollapseRow

TXGridStyle

TXGridStyle represents a drawing style for a grid.

Unit
XDBGrids

Description
TXDBGrid has an associated TXGridStyle that holds additional parameters needed to determine drawing style.

See also: TXCustomDBGrid.GridStyle

TXGridStyle.DataRowSpace

Specifies the number of empty lines in pixels for data row.

```
property DataRowSpace: Integer;
```

Description
DataRowSpace determines the number of empty lines in pixels in data row. The default value is 3 pixels for vsStandard style drawing and 4 pixels for vsXPStyle style drawing. See also VisualStyle.

See also: TXGridStyle.HeaderRowSpace, TXGridStyle.TitleRowSpace, TXGridStyle.TotalRowSpace, TXGridStyle.VisualStyle

TXGridStyle.Grid

Indicates the grid that contains this style.

```
property Grid: TXCustomDBGrid;
```

Description
Use the Grid property to access the grid that contains this style.

See also: TXCustomDBGrid

TXGridStyle.HeaderRowSpace

Specifies the number of empty lines in pixels for header row.

```
property HeaderRowSpace: Integer;
```

Description
HeaderRowSpace determines the number of empty lines in pixels in header row. The default value is 4 pixels for vsStandard style drawing and 8 pixels for vsXPStyle style drawing. See also VisualStyle.

See also: TXGridStyle.DataRowSpace, TXGridStyle.TitleRowSpace, TXGridStyle.TotalRowSpace, TXGridStyle.VisualStyle

TXGridStyle.TitleColMargin

Specifies the left and right margin in pixels for column's title.

```
property TitleColMargin: Integer;
```

Description
TitleColMargin determines the number of pixels between frame and text in column's title for left and right side. The default value is 0 for vsStandard style drawing and 1 pixel for vsXPStyle style drawing. See also VisualStyle.

See also: TXGridStyle.TitleRowSpace, TXGridStyle.VisualStyle

TXGridStyle.TitleRowSpace

Specifies the number of empty lines in pixels for title row.

```
property TitleRowSpace: Integer;
```

Description
TitleRowSpace determines the number of empty lines in pixels in title row. The default value is 4 pixels for vsStandard style drawing and 8 pixels for vsXPStyle style drawing. See also VisualStyle.

See also: TXGridStyle.HeaderRowSpace, TXGridStyle.DataRowSpace, TXGridStyle.TotalRowSpace, TXGridStyle.VisualStyle

TXGridStyle.TotalColMargin

Specifies the left and right margin in pixels for column's total cell.

```
property TotalColMargin: Integer;
```

Description
TotalColMargin determines the number of pixels between frame and text in column's total cell for left and right side. The default value is 0 for both vsStandard and vsXPStyle style drawing. See also VisualStyle.

See also: TXGridStyle.TotalRowSpace, TXGridStyle.VisualStyle

TXGridStyle.TotalRowSpace

Specifies the number of empty lines in pixels for total rows.

```
property TotalRowSpace: Integer;
```

Description
TotalRowSpace determines the number of empty lines in pixels in total rows. The default value is 4 pixels for vsStandard style drawing and 8 pixels for vsXPStyle style drawing. See also VisualStyle.

See also: TXGridStyle.HeaderRowSpace, TXGridStyle.DataRowSpace, TXGridStyle.TitleRowSpace, TXGridStyle.VisualStyle

TXGridStyle.VisualStyle

Determines the drawing style for the grid.

```
type
  TVisualStyle = (vsCustom, vsDefault, vsStandard, vsXPStyle);
property VisualStyle: TVisualStyle;
```

Description
Use VisualStyle property to specify the drawing style. The XDBGrid component has two typical styles. The vsStandard style is compatible with Borland's DBGrid. This style is suitable to non Windows XP platform. On Windows XP platform, when XDBGrid uses elements from ThemeServices to realize FixedTheme, some elements are greater. To obtain nice look on Windows XP platform, you should select vsXPStyle.

Default value for VisualStyle is vsDefault. This value determines using vsXPStyle, when the application is running on Windows XP platform and XP manifest is present and vsStandard style in other cases (non Windows XP platform nor XP manifest present).

When you select vsXPStyle value, this style (spacing) will be used on all Windows platform, even when IsGridThemed return False. When you select vsStandard value, the standard spacing will be used on all Windows platform, even when IsGridThemed return True (not recomended).

You can also prapare common style suitable for all Windows platform, when you select vsCustom visual style. Then, you can define your own values for other properties in GridStyle. Be sure, the defined values are resonable.

Notice. Since version 3.3 the default value for VisualStyle is vsXPStyle. This value is suitable for all Windows platform.

See also: TXGridStyle.DataRowSpace, TXGridStyle.HeaderRowSpace, TXGridStyle.TitleColMargin, TXGridStyle.TitleRowSpace

TXGridStyle.Assign

Copies the contents of the source grid style to a new grid style object.

```
procedure Assign(Source: TPersistent); override;
```

Description
Assign copies the published properties of the source object if it is another TXGridStyle object. Otherwise, it calls the inherited method, which allows properties to be copied from any object that implements its AssignTo method with a target of TXGridStyle.

See also: TXCustomDBGrid

TXGridStyle.Create

Creates and initializes a GridStyle object.

```
constructor Create(AGrid: TXCustomDBGrid);
```

Description
Call Create to instantiate an instance of TXGridStyle. Create takes a TXCustomDBGrid instance as its argument. You not need to create grid style directly. This is instantiated by XDBGrid.

See also: TXCustomDBGrid

TXScrollProp

TXScrollProp represents a additional possibilities for scrollbars.

Unit
XDBGrids

Description
TXDBGrid has an associated TXScrollProp that holds additional properties for scrollbars. TXScrollProp's properties are practicable only when FlatSBMode is accessible.

See also: TXCustomDBGrid.ScrollProp, TXCustomDBGrid.ScrollBars, TXCustomDBGrid.FlatSBMode

TXScrollProp.AutoHidden

Specifies whether the grid can automatically show/hide scroll bars.

```
property AutoHidden: TScrollStyle;
```

Description
Use AutoHidden to indicate whether the grid horizontal or vertical scroll bar is automatically hidden. When horizontal scroll bar is automatically hidden then, if all the cells in the grid fit in the ClientWidth, no horizontal scrollbar appears, even if ScrollBars is ssHorizontal or ssBoth. When vertical scroll bar is automatically hidden then, if all the cells fit in the ClientHeight, no vertical scrollbar appears, even if ScrollBars is ssVertical or ssBoth.

Notice. When FlatSBMode is False the AutoHidden property is of no importance.

See also: TXCustomDBGrid.ScrollBars, TXScrollProp.AutoStretch, TXScrollProp.Color, TXScrollProp.Style, TXScrollProp.ThumbBar

TXScrollProp.AutoStretch

Specifies whether the grid hides horizontal scrollbar when StretchMode is active.

```
property AutoStretch: Boolean;
```

Description
Use AutoStretch to indicate whether the horizontal scrollbar should be hidden when StretchMode is active. When AutoStretch is True (default), the horizontal scrollbar is hidden when StretchMode is active. When AutoStretch is False the horizontal scrollbar works according to other settings in ScrollProp.

See also: TXCustomDBGrid.ScrollBars, TXScrollProp.AutoHidden, TXScrollProp.Color, TXScrollProp.Style, TXScrollProp.ThumbBar

TXScrollProp.Color

Specifies the scroll bar's color.

```
property Color: TColor;
```

Description
Use Color to specify the scroll bar's color. When you explicitly assign a value to Color, the ParentColor property is automatically set to False. If the scroll bar's ParentColor property is True, then the Color property reflects the Windows button highlight color (clBtnHighlight), and changing this color in the Windows control panel automatically updates the value of Color.

Notice. When FlatSBMode is False the Color property is of no importance.

See also: TXScrollProp.ParentColor, TXScrollProp.Style, TXScrollProp.AutoHidden, TXScrollProp.ThumbBar

TXScrollProp.Grid

Indicates the grid that contains this object.

```
property Grid: TXCustomDBGrid;
```

Description
Use the Grid property to access the grid that contains this object.

See also: TXCustomDBGrid

TXScrollProp.ParentColor

Specifies whether the scroll bar should reflect the Windows button highlight color.

```
property ParentColor: Boolean;
```

Description
Use ParentColor to indicate that the scroll bar should always reflect the Windows button highlight color. Using ParentColor differs from setting the Color property to clBtnHighlight in that when ParentColor is True, changing the button highlight color in the Windows control panel automatically updates the value of Color. When the value of the Color property is changed, its ParentColor property is automatically set to False.

See also: TXScrollProp.Color

TXScrollProp.ParentStyle

Specifies whether the scroll bar style is determined by XDBGrid style.

```
property ParentStyle: Boolean;
```

Description
Use ParentStyle to indicate that the scroll bar should always reflect the XDBGrid's FixedStyle. When ParentStyle is True and FixedStyle is changed, the Style property is automatically updated to achieve best look for the grid. When the value of the Style property is changed, its ParentStyle property is automatically set to False.

See also: TXScrollProp.Style

TXScrollProp.Style

Specifies a scroll bar style.

```
property Style: TScrollBarStyle;
```

Description
Use Style to specify the scroll bar's style. The style determines whether the scroll bar appears three-dimensional, flat, or uses hot tracking. When you explicitly assign a value to Style, the ParentStyle property is automatically set to False.

Notice. When FlatSBMode is False the Style property is of no importance.

See also: TXScrollProp.ParentStyle, TXScrollProp.Color, TXScrollProp.AutoHidden, TXScrollProp.ThumbBar

TXScrollProp.ThumbBar

Specifies the size of the scroll bar's thumb tab.

```
property ThumbBar: Boolean;
```

Description
Use ThumbBar to specify the size of the scroll bar's thumb tab. When ThumbBar is True the thumb tab is a bar. When ThumbBar is False the thumb tab is a box. See also dgThumbTracing option in Options.

Notice. When FlatSBMode is False the ThumbBar property is of no importance.

See also: TXScrollProp.Color, TXScrollProp.Style, TXScrollProp.AutoHidden

TXScrollProp.WinXPMode

Specifies whether the grid uses flat scrollbar on Windows XP platform.

```
property WinXPMode: Boolean;
```

Description
WinXPMode determines whether the grid uses flat scroll bars when the application is running on Windows XP platform and Manifest is'nt present. In this case, when WinXPMode is True, the XP style scroll bars appear by default. When you set WinXPMode to False, you can see flat scroll bars even on Windows XP platform. This possibility is especially useful for developers of Delphi 7 and above. When you remove manifest resource and/or manifest file from application, the FlatSBMode will return True even on Windows XP platform and then, you can test flat scroll bars without running application on non-Windows XP platforms.

When Manifest is present the FlatSBMode is always False on Windows XP platform.

See also: TXCustomDBGrid.ScrollBars, TXCustomDBGrid.FlatSBMode

TXScrollProp.Create

Creates and initializes a ScrollProp object.

```
constructor Create(AGrid: TXCustomDBGrid);
```

Description
Call Create to instantiate an instance of TXScrollProp. Create takes a TXCustomDBGrid instance as its argument. You not need to create ScrollProp object directly. This is instantiated by XDBGrid.

See also: TXCustomDBGrid

TXDBGridSettings

TXDBGridSettings represents a layout management for data grid (TXDBGrid).

Unit
XDBGrids

Description
TXDBGrid has an associated TXDBGridSettings that holds additional properties for columns layout management. They allow to save/load columns layout (width, order, visibility, etc.) to/from registry or *.ini file.

See also: TXCustomDBGrid.Settings

TXDBGridSettings.Active

Specifies whether or not the grid settings are stored and retrieved automatically.

```
property Active: Boolean;
```

Description
Use Active to determine or set whether the grid settings are stored and retrieved from ini file or from registry automatically. When Active is False the grid cannot load or save Layout automatically, but the developer can call directly LoadLayout and SaveLayout methods in the code. When Active is True, Layout settings can be load and save automatically according to soLoadLayout/soSaveLayout Options. The Layout settings are loaded automatically when DataSource linked to XDBGrid is first time opened and saved when XDBGrid is destroyed.

See also: TXDBGridSettings.Layout, TXDBGridSettings.IniFile, TXDBGridSettings.Options, TXDBGridSettings.LoadLayout, TXDBGridSettings.SaveLayout

TXDBGridSettings.DefaultFileName

Specifies default name of the ini file or default name of the registry key.

```
const
  DefaultIniFileName: string = '';
  DefaultRegistryKey: string = '\Software\X-Files Software\XDBGrid';
  DefaultRegistryKey: string = '\Software\X-Files
Software\(Application.ExeName)'; {* ver. 8.0 *}

property DefaultFileName: string;
```

Description
When IniFile is True, the DefaultFileName property retrieves default FileName from `DefaultIniFileName` global variable. You can initialize DefaultIniFileName variable in the begining of the project file as common ini file name for all instances of XDBGrid in the project. By default, DefaultIniFileName variable is empty. When DefaultIniFileName variable is empty the DefaultFileName property return name of executable application with an .INI extension as default FileName (e.g. `'PROJECT1.INI'`).

When IniFile is False, the DefaultFileName property retrieves default registry key from `DefaultRegistryKey` global variable. You can initialize DefaultRegistryKey variable in the begining of the project file as common registry key for all instances of XDBGrid in the project. By default, DefaultRegistryKey variable is initialized by `'\Software\X-Files Software\XDBGrid'` value. You can change this value, but the variable can not be empty. Remember that TRegistryIniFile reinterprets the FileName property as a subkey under the system registry's root key (HKEY_CURRENT_USER by default).

Since version 8.0 new ExeName property was added (default True). When ExeName is True and original (unchanged) DefaultRegistryKey is used, the Application.ExeName replaces "XDBGrid" subkey name. Set ExeName property to False in older applications to load/save Layout from/to old place. `{* ver. 8.0 *}`

See also: TXDBGridSettings.IniFile, TXDBGridSettings.DefaultSection, TXDBGridSettings.FileName, TXDBGridSettings.ExeName

TXDBGridSettings.DefaultSection

Specifies default name of the section in ini file or in the registry key.

```
property DefaultSection: string;
```

Description

The DefaultSection property retrieves default name of the section in ini file or in the registry key. Default name of the section is composed of parent form name + '.' + grid name (e.g. `'Form1.XDBGrid1'`).

See also: TXDBGridSettings.Section, TXDBGridSettings.DefaultFileName

TXDBGridSettings.ExeName

Specifies whether the default registry key contains name of application.

```
property ExeName: Boolean; {* ver. 8.0 *}
```

Description

When ExeName is True and original (unchanged) DefaultRegistryKey is used to load/save Layout from/to registry, the Application.ExeName replaces "XDBGrid" subkey name. Set ExeName to False in older applications to load/save Layout from/to old place.

See also: TXDBGridSettings.Layout, TXDBGridSettings.DefaultFileName

TXDBGridSettings.FileName

Contains the name of the ini file or the name of the registry key.

```
property FileName: string;
```

Description

When IniFile is True, the FileName property should contain the name of the ini file to save/load columns Layout. When IniFile is False, the FileName property should contain then name of the registry key to save/load columns layout.

The FileName property is passed as parameter when TIniFile or TRegistryIniFile object is created to save/load columns layout. When FileName property is empty the value of DefaultFileName property will be used as default.

See also: TXDBGridSettings.IniFile, TXDBGridSettings.Section, TXDBGridSettings.DefaultFileName

TXDBGridSettings.Grid

Identifies the grid that owns this object.

```
property Grid: TXCustomDBGrid;
```

Description
Use the Grid property to access the grid that owns this object.

See also: TXCustomDBGrid

TXDBGridSettings.IniFile

Specifies whether the grid should save/load layout either to/from *.ini file or to/from registry.

```
property IniFile: Boolean;
```

Description
Use IniFile to indicate whether the grid should save/load columns Layout either to/from *.ini file (when IniFile = True) or to/from registry (when IniFile = False).

See also: TXDBGridSettings.Layout, TXDBGridSettings.FileName

TXDBGridSettings.Layout

Contains the individual lines of text for column layout settings.

```
property Layout: TStrings;
```

Description
The Layout is a TStrings object designed for store and retrieve layout settings as name-value pairs. For more information on name-value pairs, refer to Values property.

The Layout object contains several lines with column settings stored into Columns.Settings property and optionally few additional including FilterGrid.Settings property.

When soStretchMode, soOrderFields, soLinesCount, ... are included in Options , the additional lines with names "StretchMode", "OrderFields", "LinesCount" are added to Layout.

When soUserSettings option is included in Options, the additional lines "AutoLoadLayout" and "AutoSaveLayout" are added to Layout. You can also create OnLayout event handler to add your custom lines (properties or options) to the Layout.

This convention corresponds to the format of configuration files (*.ini). For example (gray lines are optional):

```
[MainForm.CustomerDBGrid]
AutoLoadLayout=1
AutoSaveLayout=1
Columns=9
Column1=CustNo,80,1,0,1
Column2=Company,170,0,0,1
Column3=Country,125,1,0,1
Column4=State,125,1,0,1
Column5=Zip,69,1,0,1
Column6=City,103,1,0,1
Column7=Addr1,138,1,0,1
Column8=Phone,120,1,0,1
Column9=FAX,120,1,0,1
StretchMode=0
CurrentPPI=120
OrderFields=Country DESC,State,City
LinesCount=1
MultiSelect=0
SelectedRow=1
StripedRows=0
AutoFilter=1
Filtered=1
Filters=2
Filter1=Country,Country LIKE '%US%',
Filter2=State,,AL,CA,FL,OR,TX
```

The Layout object is designed to store and retrieve grid layout to/from ini file or to/from registry but you can also use TStrings methods like: LoadFromFile/SaveToFile, LoadFromStream/SaveToStream, GetTextStr/SetTextStr (or Text property) to load/save Layout to the text file, any stream (BLOB Field) or memory variable. Each time you are reading Layout property the current column settings are retrieved from XDBGrid's properties. Each time you are setting new Layout property the XDBGrid's columns properties are modified. After you load Settings.Text property from the text file, stream or variable you need to call ApplyLayout 0 method to apply changes to the grid columns.

See also: TXDBGridSettings.Active, TXDBGridSettings.IniFile, TXDBGridSettings.Options, TXDBGridSettings.LoadLayout, TXDBGridSettings.SaveLayout, TXCustomDBGrid.OnLayout

TXDBGridSettings.Options

Specifies various behavioral properties of the XDBGrid settings.

```
type
  TXSettingsOption = (soLoadLayout, soSaveLayout, soUserSettings,
    soStretchMode, soOrderFields, soLinesCount,
    soMultiSelect, soSelectedRow, soStripedRows, soAutoFilter, soFiltered,
    soFilters); {* ver. 8.0 *}
  TXSettingsOptions = set of TXSettingsOption;
property Options: TXSettingsOptions;
```

Description
Set Options to include the desired properties for the XDBGrid settings. Options is a set drawn from the following values:

Value	Meaning
soLoadLayout	Load Layout from ini file or from registry always when DataSource linked to XDBGrid is first time opened.
soSaveLayout	Save Layout to ini file or to registry always when XDBGrid is destroyed.
soUserSettings	Include soUserSettings in Options when soLoadLayout/soSaveLayout option can be changed by user in run-time and should also be stored and retrieved form ini file or from registry.
soStretchMode	Add StretchMode property to Layout.
soOrderFields	Add OrderFields property to Layout.
soLinesCount	Add LinesCount property to Layout.
soMultiSelect	Add dgMultiSelect option to Layout.
soSelectedRow	Add gdoSelectedRow option to Layout.
soStripedRows	Add gdoStripedRows option to Layout.
soAutoFilter	Add FilterGrid.AutoFilter property to Layout.
soFiltered	Add FilterGrid.Filtered property to Layout.
soFilters	Add FilterGrid.Settings property to Layout.

See also: TXDBGridSettings.Layout, TXDBGridSettings.IniFile, TXDBGridSettings.LoadLayout, TXDBGridSettings.SaveLayout

TXDBGridSettings.Section

Contains the name of the section in ini file or in the registry.

```
property Section: string;
```

Description
When IniFile is True, the Section property should contain the name of the section in FileName ini file to save/load columns Layout. When IniFile is False, the Section property should contain then name of the section in FileName registry key to save/load columns layout.

The Section property is passed as parameter when any method of TIniFile or TRegistryIniFile is used to save/load columns layout. When Section property is undefined the value of DefaultSection property is used as default. The Section property can not be empty.

See also: TXDBGridSettings.FileName, TXDBGridSettings.DefaultSection

TXDBGridSettings.ApplyLayout

Applies the column layout stored in Layout buffer.

```
procedure ApplyLayout;
```

Description
Use ApplyLayout to apply current Layout buffer in the grid. This procedure is useful after you load new layout to Layout buffer by calling ReadLayout, LoadFromFile, LoadFromStream methods or after you setting new value for Text property.

Notice: When value of "Columns" name-value pair in the Layout buffer does not correspond to Count of Columns in the XDBGrid the columns in the grid are not changed.

See also: TXDBGridSettings.Layout, TXDBGridSettings.ReadLayout, TXDBGridSettings.LoadLayout

TXDBGridSettings.Create

Creates and initializes a Settings object.

```
constructor Create(AGrid: TXCustomDBGrid);
```

Description
Call Create to instantiate an instance of TXDBGridSettings. Create takes a TXCustomDBGrid instance as its argument. You not need to create Settings object directly. This is instantiated by XDBGrid.

See also: TXCustomDBGrid, TXDBGridSettings.Destroy

TXDBGridSettings.Destroy

Destroys an instance of TXDBGridSettings.

```
destructor Destroy; override;
```

Description
Destroy frees the helper objects of the TXDBGridSettings before destroying the instance. The user never need to call this method directly.

See also: TXCustomDBGrid, TXDBGridSettings.Create

TXDBGridSettings.EraseLayout

Clears the column layout into ini file.

```
procedure EraseLayout(IniFile: TCustomIniFile = nil);
```

Description
Use EraseLayout to clear the column layout stored into IniFile by using SaveLayout. You can specify the existing IniFile object as the parameter.

When IniFile parameter is empty the default IniFile object will be created to perform EraseLayout and the default IniFile will be destroyed after this operation. Default IniFile is created as TIniFile or TRegistryIniFile according to current IniFile and FileName property.

See also: TXDBGridSettings.Layout, TXDBGridSettings.LayoutExists, TXDBGridSettings.SaveLayout

TXDBGridSettings.LayoutExists

Check the column layout exists into ini file.

```
function LayoutExists(IniFile: TCustomIniFile = nil): Boolean;
```

Description
Use LayoutExists to check the column layout stored into IniFile by using SaveLayout exists. You can specify the existing IniFile object as the parameter.

When IniFile parameter is empty the default IniFile object will be created to perform LayoutExists and the default IniFile will be destroyed after this operation. Default IniFile is created as TIniFile or TRegistryIniFile according to current IniFile and FileName property.

See also: TXDBGridSettings.Layout, TXDBGridSettings.EraseLayout, TXDBGridSettings.SaveLayout

TXDBGridSettings.LoadLayout

Loads and applies the column layout from ini file.

```
procedure LoadLayout(IniFile: TCustomIniFile = nil);
```

Description
Use LoadLayout to load the column layout from IniFile into Layout buffer and to apply the new layout in the grid. You can specify the existing IniFile object as the parameter.

When IniFile parameter is empty the default IniFile object will be created to perform LoadLayout and the default IniFile will be destroyed after this operation. Default IniFile is created as TIniFile or TRegistryIniFile according to current IniFile and FileName property.

Notice: When value of "Columns" name-value pair in the Layout buffer does not correspond to Count of Columns in the XDBGrid the columns in the grid are not changed.

See also: TXDBGridSettings.Layout, TXDBGridSettings.SaveLayout, TXDBGridSettings.ReadLayout

TXDBGridSettings.ReadLayout

Loads the column layout from ini file to Layout buffer.

```
function ReadLayout(IniFile: TCustomIniFile = nil): TStrings;
```

Description
Use ReadLayout to load the column layout from IniFile into Layout buffer. The new layout is'nt apply in the grid. You can modify the loaded layout (result of ReadLayout function) before you call ApplyLayout to apply new layout in the grid. You can specify the existing IniFile object as the parameter.

When IniFile parameter is empty the default IniFile object will be created to perform ReadLayout and the default IniFile will be destroyed after this operation. Default IniFile is created as TIniFile or TRegistryIniFile according to current IniFile and FileName property.

See also: TXDBGridSettings.Layout, TXDBGridSettings.ApplyLayout, TXDBGridSettings.LoadLayout

TXDBGridSettings.SaveLayout

Saves the column layout to ini file.

```
procedure SaveLayout(IniFile: TCustomIniFile = nil);
```

Description
Use SaveLayout to save the column layout to IniFile. You can specify the existing IniFile object as the parameter.

When IniFile parameter is empty the default IniFile object will be created to perform SaveLayout and the default IniFile will be destroyed after this operation. Default IniFile is created as TIniFile or TRegistryIniFile according to current IniFile and FileName property.

See also: TXDBGridSettings.Layout, TXDBGridSettings.LoadLayout, TXDBGridSettings.ReadLayout, TXDBGridSettings.EraseLayout, TXDBGridSettings.LayoutExists

TXDBGridSettings.StoreLayout

Store the current content of Layout buffer to ini file.

```
procedure StoreLayout(IniFile: TCustomIniFile = nil); {* ver. 8.0 *}
```

Description
Use StoreLayout to save the current content of Layout buffer to IniFile. You can specify the existing IniFile object as the parameter.

When IniFile parameter is empty the default IniFile object will be created to perform StoreLayout and the default IniFile will be destroyed after this operation. Default IniFile is created as TIniFile or TRegistryIniFile according to current IniFile and FileName property.

See also: TXDBGridSettings.Layout, TXDBGridSettings.ReadLayout, TXDBGridSettings.SaveLayout

TXDBGridSettings.SwitchOptions

Include or exclude Options depending on State.

```
procedure SwitchOptions(Part: TXSettingsOptions; State: Boolean); {* ver. 6.0 *}
```

Description
Use SwitchOptions to include or exclude the Part of Options property. When State is True the Part options are included, otherwise excluded.

See also: TXCustomDBGrid.SwitchOptions

TXDBGridGradient

TXDBGridGradient specifies the gradient drawing style for grid.

Unit
XDBGrids

Description
TXDBGridGradient class is derived from TXFGradientButton class to realize gradient drawing for grid's fixed and data cells. `{* ver. 5.0 *}`

See also: TXCustomDBGrid.Gradient

TXBookmarkList

TXBookmarkList is a collection of bookmarks that identify a set of records in a dataset.

Unit
XDBGrids

Description
Use TXBookmarkList to manage a set of bookmarks when working with the records in a dataset. Each bookmark identifies a particular record in the dataset, and can be identified by a string.

See also: TXCustomDBGrid.SelectedRows

TXBookmarkList.Count

Indicates the number of bookmarks in the bookmark list.

```
property Count: Integer;
```

Description
Read Count to determine the number of entries in the Items array. Use Count as an upper limit when iterating over the strings in the Items array.

See also: TXBookmarkList.Items

TXBookmarkList.CurrentRowSelected

Indicates whether the Bookmark property of the associated XDBGrid's dataset is in the Items array.

```
property CurrentRowSelected: Boolean;
```

Description
Read CurrentRowSelected to determine whether the Bookmark property of the associated XDBGrid's dataset specifies a bookmark string in the Items property array. CurrentRowSelected is True when the bookmark string is included, False when it is not. The Bookmark property of the dataset indicates the current record in the dataset.

Set CurrentRowSelected to specify whether the Bookmark string of the associated XDBGrid's dataset should be included in the bookmark list. Setting CurrentRowSelected to True adds the Bookmark property of the dataset to the Items array if it is not already there. Setting CurrentRowSelected to False removes the Bookmark of the dataset from the Items array if it is there. If the Bookmark property of the dataset is an empty string (no bookmark), setting CurrentRowSelected does nothing.

Note: Trying to read or write CurrentRowSelected when the dataset is not active will raise an EInvalidGridOperation exception.

See also: TXBookmarkList.Items

TXBookmarkList.Items

Provides indexed access to the bookmarks for marked records in the dataset.

```
property Items[Index: Integer]: TBookmarkStr; default;
property Items[Index: Integer]: TBookmark; default; // Delphi 2009, 2010, XE
```

Description
Use Items to obtain a bookmark string from the set maintained by the TXBookmarkList object. Each bookmark string represents a single record in the dataset of the associated XDBGrid. Bookmark strings can be compared with other bookmark strings to determine relative position within the database, and used to position the dataset on an arbitrary record. The Index parameter indicates the index of the bookmark, where 0 is the index of the first bookmark, 1 is the index of the second bookmark, and so on.

Use Items with the Count property to iterate through all of the bookmarks in the list.

Note: Bookmarks in the bookmark list are not automatically deleted when records in the dataset are deleted. As a result, some bookmarks in the Items array may become invalid. Call the Refresh method to delete invalid bookmarks from the Items array.

See also: TXBookmarkList.Clear, TXBookmarkList.Count, TXBookmarkList.Find, TXBookmarkList.Refresh

TXBookmarkList.OrAllRows

Specifies "proxy" class that returns enumerator for selected rows or all data rows in the grid.

```
property OrAllRows: TXBookmarkListOrAllRows; {* ver. 5.6 *} // For Delphi 2009
or higher
```

Description
Use OrAllRows only when you need enumerator for selected rows or all data rows in the grid. This enumerator is similar to GetEnumerator except when no selected rows in bookmark list. In that case the enumerator iterates over all data rows in the grid.

```
type
  Bookmark: TBookmark;
begin
  for Bookmark in XDBGrid1.SelectedRows.OrAllRows do
  begin
    // Perform action on each record selected in XDBGrid1.SelectedRows, but
when no selected rows, perform action on each record into
XDBGrid1.DataSource.DataSet
  end;
end;
```

See also: TXBookmarkList.GetEnumerator, TXBookmarkList.OrCurrent

TXBookmarkList.OrCurrent

Specifies "proxy" class that returns enumerator for selected rows or current row in the grid.

```
property OrCurrent: TXBookmarkListOrCurrent; {* ver. 5.6 *} // For Delphi 2009 or higher
```

Description

Use OrCurrent only when you need enumerator for selected rows or current row in the grid. This enumerator is similar to GetEnumerator except when no selected rows in bookmark list. In that case the enumerator returns only one (current) row.

```
type
  Bookmark: TBookmark;
begin
  for Bookmark in XDBGrid1.SelectedRows.OrCurrent do
  begin
    // Perform action on each record selected in XDBGrid1.SelectedRows, but when no selected rows, perform action on current record into XDBGrid1.DataSource.DataSet
  end;
end;
```

See also: TXBookmarkList.GetEnumerator, TXBookmarkList.OrAllRows

TXBookmarkList.Clear

Deletes all bookmarks from the list.

```
procedure Clear;
```

Description

Call Clear to empty the Items array and set the Count property to 0 when all rows of the associated XDBGrid are deselected. Calling Clear invalidates the associated XDBGrid so that the selected rows that were referenced by the bookmarks in the bookmark list can be repainted.

Note: Clear removes the bookmarks from the bookmark list to reflect the fact that the corresponding rows are not selected. To delete the rows from the dataset rather than just deselecting them, call the Delete method instead.

See also: TXBookmarkList.Count, TXBookmarkList.Delete, TXBookmarkList.Items, TXBookmarkList.Refresh

TXBookmarkList.Create

Creates an instance of TXBookmarkList and associates it with a XDBGrid.

```
constructor Create(AGrid: TXCustomDBGrid);
```

Description

Applications do not need to call create to instantiate the bookmark list used by XDBGrids to implement their SelectedRows property. XDBGrids call Create from their constructor to create this bookmark list. Component writers that implement additional properties for descendants of TXCustomDBGrid can create additional bookmark list objects to represent other sets of records in the dataset.

See also: TXBookmarkList.Destroy, TXCustomDBGrid.SelectedRows

TXBookmarkList.Delete

Deletes all the records specified by the bookmarks in the list from the dataset of the associated XDBGrid.

```
procedure Delete;
```

Description
Call Delete to delete the set of records represented by the bookmark list from the dataset. To remove all the bookmarks from the list without changing the dataset, use the Clear method instead.

See also: TXBookmarkList.Clear, TXBookmarkList.Items

TXBookmarkList.Destroy

Destroys an instance of TXBookmarkList.

```
destructor Destroy; override;
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TXBookmarkList object is not nil, and only then calls Destroy.

Destroy frees the memory used to store the Items property array.

See also: TXBookmarkList.Create, TXBookmarkList.Items

TXBookmarkList.Find

Indicates whether a specified bookmark string is contained in the bookmark list.

```
function Find(const Item: TBookmarkStr; var Index: Integer): Boolean;
function Find(const Item: TBookmark; var Index: Integer): Boolean; // Delphi 2009, 2010, XE
```

Description
Call Find to determine whether the string specified by the Item parameter is one of the bookmark strings in the Items property array. Find returns True if the bookmark list contains the bookmark string, False otherwise. If Find returns True, the index of the bookmark string is returned as the value of the Index parameter.

Note: The index of a specific bookmark string can also be obtained by using the IndexOf method. Unlike the IndexOf method, which returns a value of -1 if the bookmark string is not in the Items array, Find does not change the value of the Index parameter if the bookmark string can't be found. Use Find when a default value should be changed only if the bookmark is not in the list.

See also: TXBookmarkList.IndexOf, TXBookmarkList.Items

TXBookmarkList.GetEnumerator

Creates enumerator for TXBookmarkList class.

```
function GetEnumerator: TXBookmarkListEnumerator; {* ver. 5.6 *} // For Delphi 2009 or higher
```

Description
You not need to call this function directly. It allows you to iterate over all selected rows in the grid.

```
type
  Bookmark: TBookmark;
begin
  for Bookmark in XDBGrid1.SelectedRows do
  begin
    // Perform action on each record selected in XDBGrid1.SelectedRows
  end;
end;
```

See also: TXCustomDBGrid.DataRows, TXBookmarkList.OrAllRows, TXBookmarkList.OrCurrent

TXBookmarkList.IndexOf

Returns the index of a bookmark string in the Items property array.

```
function IndexOf(const Item: TBookmarkStr): Integer;
function IndexOf(const Item: TBookmark): Integer; // Delphi 2009, 2010, XE
```

Description
Call IndexOf to get the index for a bookmark string in the Items array. The first item in the array has index 0, the second item has index 1, and so on. If an item is not in the list, IndexOf returns -1.

See also: TXBookmarkList.Find, TXBookmarkList.Items

TXBookmarkList.Refresh

Removes all invalid bookmarks from the Items property array.

```
function Refresh: Boolean;
```

Description
Bookmarks in the bookmark list are not automatically deleted when records in the dataset are deleted. As a result, some bookmarks in the Items array may become invalid. Call Refresh to ensure that every bookmark in the bookmark list is valid. Refresh attempts to locate every record in the dataset that corresponds to a bookmark in the Items property array. All bookmarks that do not correspond to records in the dataset are removed.

If any of the bookmarks in the list are invalid, Refresh returns True, and invalidates the XDBGrid so that it will repaint, reflecting the removal of all invalid records. If all bookmarks in the list are still valid, Refresh returns False.

See also: TXBookmarkList.Items

TXBookmarkList.Restore

Restore bookmarks from the saved list of key fields.

```
procedure Restore(AKeyFields: string = ''); {* ver. 5.32 *}
```

Description
Call Restore to restore bookmarks from the list of record's key fields saved by Save method. AKeyFields parameter specifies the field or fields to uniquely identify records in dataset. To use more than one field, separate each field name with a semicolon. When AKeyFields parameter is empty, the KeyFields property will be used. When KeyFields property is empty, an error will be raised. The specified key fields must be of the same types as the corresponding key fields in Save method, or the Restore method fails.

See also: TXBookmarkList.Save

TXBookmarkList.Save

Save bookmarks to the list of key fields.

```
procedure Save(AKeyFields: string = ''); {* ver. 5.32 *}
```

Description
Call Save to save bookmarks to the list of record's key fields. AKeyFields parameter specifies the field or fields to uniquely identify records in dataset. To use more than one field, separate each field name with a semicolon. When AKeyFields parameter is empty, the KeyFields property will be used. When KeyFields property is empty, an error will be raised. Each call of Save procedure overrides the saved list of key fields.

See also: TXBookmarkList.Restore

TXColumnList

TXColumnList is a collection of columns that are selected in the grid.

Unit
XDBGrids

Description
Use TXColumnList to manage a set of columns that are selected in the grid.

See also: TXColumnList

TXColumnList.Columns

Indicates whether any column of the XDBGrid is in the Items array (column is selected).

```
property Columns[Index: Integer]: Boolean; {* ver. 4.3 *}
```

Description
Read Columns for specified Index to determine whether the column of the XDBGrid specifies a column in the Items property array. Columns[Index] is True when the column is included (selected), False when it is not.

Set Columns[Index] to specify whether the column of the XDBGrid should be included in the column list. Setting Columns[Index] to True adds the column of the grid for specifies Index to the Items array if it is not already there. Setting Columns[Index] to False removes the column of the grid for specified Index from the Items array if it is there.

See also: TXColumnList.CurrentColSelected, TXColumnList.Items

TXColumnList.Count

Indicates the number of columns in the column list.

```
property Count: Integer; {* ver. 4.3 *}
```

Description
Read Count to determine the number of entries in the Items array. Use Count as an upper limit when iterating over the columns in the Items array.

See also: TXColumnList.Items

TXColumnList.CurrentColSelected

Indicates whether the SelectedColumn property of the XDBGrid is in the Items array.

```
property CurrentColSelected: Boolean; {* ver. 4.3 *}
```

Description
Read CurrentColSelected to determine whether the SelectedColumn property of the XDBGrid specifies a column in the Items property array. CurrentColSelected is True when the column is included, False when it is not. The SelectedColumn property of the XDBGrid indicates the current column in the grid.

Set CurrentColSelected to specify whether the SelectedColumn of the XDBGrid should be included in the column list. Setting CurrentColSelected to True adds the SelectedColumn property of the grid to the Items array if it is not already there. Setting CurrentColSelected to False removes the SelectedColumn of the grid from the Items array if it is there. If the SelectedColumn property of the grid is nil, setting CurrentColSelected does nothing.

See also: TXColumnList.Items

TXColumnList.Items

Provides indexed access to the selected columns in the grid.

```
property Items[Index: Integer]: TXColumn; default; {* ver. 4.3 *}
```

Description
Use Items to obtain a column from the set maintained by the TXColumnList object. Each item represents a single selected column in the XDBGrid. Columns can be compared with other columns to determine relative position within the grid. The Index parameter indicates the index of the column in the list, where 0 is the index of the first selected column, 1 is the index of the second selected column, and so on.

Use Items with the Count property to iterate through all of the columns in the list.

See also: TXColumnList.Clear, TXColumnList.Count, TXColumnList.Find

TXColumnList.OrAllCols

Specifies "proxy" class that returns enumerator for selected columns or all columns in the grid.

```
property OrAllCols: TXColumnListOrAllCols; {* ver. 5.6 *} // For Delphi 2009 or higher
```

Description
Use OrAllCols only when you need enumerator for selected columns or all columns in the grid. This enumerator is similar to GetEnumerator except when no selected columns in column list. In that case the enumerator iterates over all columns in the grid.

```
type
  Column: TXColumn;
begin
  for Column in XDBGrid1.SelectedCols.OrAllCols do
  begin
    // Perform action on each column selected in XDBGrid1.SelectedCols, but when no selected columns, perform action on all columns in XDBGrid1.Columns
  end;
end;
```

See also: TXColumnList.GetEnumerator, TXColumnList.OrCurrent

TXColumnList.OrCurrent

Specifies "proxy" class that returns enumerator for selected columns or current column in the grid.

```
property OrCurrent: TXColumnListOrCurrent; {* ver. 5.6 *} // For Delphi 2009 or higher
```

Description
Use OrCurrent only when you need enumerator for selected columns or current column in the grid. This enumerator is similar to GetEnumerator except when no selected columns in column list. In that case the enumerator returns only one (current) column.

```
type
  Column: TXColumn;
begin
  for Column in XDBGrid1.SelectedCols.OrCurrent do
  begin
```



```
    // Perform action on each column selected in XDBGrid1.SelectedCols, but
when no selected columns, perform action on current column in XDBGrid1
  end;
end;
```

See also: TXColumnList.GetEnumerator, TXColumnList.OrAllCols

TXColumnList.Clear

Deletes all columns from the list.

```
procedure Clear; {* ver. 4.3 *}
```

Description
Call Clear to empty the Items array and set the Count property to 0 when all columns of the XDBGrid are deselected. Calling Clear invalidates the associated XDBGrid so that the selected columns that were referenced in the column list can be repainted.

Note: Clear removes the columns from the column list only to reflect the fact that the columns are not selected.

See also: TXColumnList.Items, TXColumnList.Count

TXColumnList.Create

Creates an instance of TXColumnList and associates it with a XDBGrid.

```
constructor Create(AGrid: TXCustomDBGrid); {* ver. 4.3 *}
```

Description
Applications do not need to call create to instantiate the column list used by XDBGrids to implement their SelectedCols property. XDBGrids call Create from their constructor to create this column list. Component writers that implement additional properties for descendants of TXCustomDBGrid can create additional column list objects to represent other sets of columns in the grid.

See also: TXColumnList.Destroy, TXCustomDBGrid.SelectedCols

TXColumnList.Destroy

Destroys an instance of TXColumnList.

```
destructor Destroy; override; {* ver. 4.3 *}
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TXColumnList object is not nil, and only then calls Destroy.

Destroy frees the memory used to store the Items property array.

See also: TXColumnList.Create, TXColumnList.Items

TXColumnList.Find

Indicates whether a specified column is contained in the column list.

```
function Find(const Item: TXColumn; var Index: Integer): Boolean; {* ver. 4.3 *}
```

Description

Call Find to determine whether the column specified by the Item parameter is one of the column in the Items property array. Find returns True if the column list contains the column, False otherwise. If Find returns True, the index of the column is returned as the value of the Index parameter.

Note: The index of a specific column can also be obtained by using the IndexOf method. Unlike the IndexOf method, which returns a value of -1 if the column is not in the Items array, Find does not change the value of the Index parameter if the column can't be found. Use Find when a default value should be changed only if the column is not in the list.

See also: TXColumnList.IndexOf, TXColumnList.Items

TXColumnList.GetEnumerator

Creates enumerator for TXColumnList class.

```
function GetEnumerator: TXColumnListEnumerator; {* ver. 5.6 *} // For Delphi 2009 or higher
```

Description

You not need to call this function directly. It allows you to iterate over all selected columns in the grid.

```
type
  Column: TXColumn;
begin
  for Column in XDBGrid1.SelectedCols do
  begin
    // Perform action on each column selected in XDBGrid1.SelectedCols
  end;
end;
```

See also: TXDBGridColumns.GetEnumerator, TXColumnList.OrAllCols, TXColumnList.OrCurrent

TXColumnList.IndexOf

Returns the index of a column in the Items property array.

```
function IndexOf(const Item: TXColumn): Integer; {* ver. 4.3 *}
```

Description

Call IndexOf to get the index for a column in the Items array. The first item in the array has index 0, the second item has index 1, and so on. If an item is not in the list, IndexOf returns -1.

See also: TXColumnList.Find, TXColumnList.Items

TXColumnList.Restore

Restore column list from the saved list.

```
procedure Restore; {* ver. 5.32 *}
```

Description
Call Restore to restore column list from the list saved by Save method.

See also: TXColumnList.Save

TXColumnList.Save

Save column list to separate list.

```
procedure Save; {* ver. 5.32 *}
```

Description
Call Save to save column list to separate list. Each call of Save procedure overrides the saved list.

See also: TXColumnList.Restore

DelayInterval variable

Determines the amount of time, in milliseconds, that passes before grid triggers some events.

Unit
XDBGrids

```
const
  DelayInterval: Integer = 200;
```

Description
When dgDelayUpdateSequence, dgDelayUpdateTotals or dgDelaySelectedRows option is included in OptionsEx the call of UpdateSequence, UpdateTotals or SelectedRowsChanged methods is delayed. DelayInterval determines the amount of delay time. It allows to minimize the number of calls.

See also: TXCustomDBGrid.OptionsEx

NullValueString variable

Specifies string value of Null variants.

Unit
XDBGrids

```
const
  NullValueString: string = '';
```

Description
NullValueString specifies string that apperas in total cells when result of CalcValue function is Null. By default, NullValueString is the empty string.

See also: TXColumnTotalValue.NullValue

MaxDropDownRows variable

Specifies the default maximum number of rows displayed in the column's drop-down list.

Unit
XDBGrids

```
const
  MaxDropDownRows: Integer = 20; {* ver. 4.3 *}
```

Description
MaxDropDownRows determines the default maximum number of rows displayed in the drop-down list associated with the column. When DropDownRows property is 0 (default) and drop-down list has more then MaxDropDownRows items, then MaxDropDownRows items is displayed in the drop-down list.

See also: TXColumn.DropDownRows, MaxDropDownWidth

MaxDropDownWidth variable

Specifies the default maximum width of the column's drop-down list.

Unit
XDBGrids

```
const
  MaxDropDownWidth: Integer = 800; {* ver. 4.3 *}
```

Description
MaxDropDownWidth determines the default maximum width of the drop-down list associated with the column. When DropDownWidth property is 0 (default) and auto-calculated width of drop-down list is greater then MaxDropDownWidth, then displayed width of drop-down list is limited to MaxDropDownWidth.

See also: TXColumn.DropDownWidth, MaxDropDownRows

ListDropDownDelay variable

Determines the amount of time, in milliseconds, that passes before each drop-down list's item is displayed.

Unit
XDBGrids

```
const
  ListDropDownDelay: Integer = 8; {* ver. 4.3 *}
```

Description
When ListDropDownDelay is greater then 0 then display of drop-down list is animated and the value determines the amount of time beetwen each drop-down list's item is displayed.

When ListDropDown is 0 the whole drop-down list is displayed at once.

See also: TXColumn.ListOptions

ListMouseTracking variable

Specifies the mouse tracking functionality for drop-down list

Unit
XDBGrids

```
const
  ListMouseTracking: Boolean = True; {* ver. 4.3 *}
```

Description
When ListMouseTracking is True the mouse tracking functionality is available for drop-down list.

When ListMouseTracking is False this functionality is not available for drop-down list.

Note. This functionallity is available only when dgGridStyleLists option is included in OptionsExt.

See also: TXColumn.ListOptions, TXCustomDBGrid.OptionsExt

DefAutoNumberFormat variable

Specifies the default format for AutoNumber values.

Unit
XDBGrids

```
const
  DefAutoNumberFormat: string = '%d.'; {* ver. 4.3 *}
```

Description
The DefAutoNumberFormat specifies the default format for AutoNumber values. The format is using by AutoNumberStr function.

See also: TXColumn.AutoNumber, AutoNumberStr

DefAutoNumberTitle variable

Specifies the default title for AutoNumber column.

Unit
XDBGrids

```
const
  DefAutoNumberTitle: string = 'No.'; {* ver. 4.3 *}
```

Description
When AddAutoNumber function is using the DefAutoNumberTitle value initializes Title.Caption for the new column.

See also: TXDBGridColumns.AddAutoNumber, TColumnTitle.Caption, DefAutoNumberWidth

DefAutoNumberWidth variable

Specifies the default width for AutoNumber column.

Unit
XDBGrids

```
const
  DefAutoNumberWidth: Integer = 48; {* ver. 4.3 *}
```

Description
When AddAutoNumber function is using the DefAutoNumberWidth value initializes Title.Width for the new column.

See also: TXDBGridColumns.AddAutoNumber, TColumn.Width, DefAutoNumberTitle

DefAutoSelectTitle variable

Specifies the default title for AutoSelect column.

Unit
XDBGrids

```
const
  DefAutoSelectTitle: string = 'Select'; {* ver. 4.3 *}
```

Description
When AddAutoSelectfunction is using the DefAutoSelectTitle value initializes Title.Caption for the new column.

See also: TXDBGridColumns.AddAutoSelect, TColumnTitle.Caption, DefAutoSelectWidth

DefAutoSelectWidth variable

Specifies the default width for AutoSelect column.

Unit
XDBGrids

```
const
  DefAutoSelectWidth: Integer = 48; {* ver. 4.3 *}
```

Description
When AddAutoSelect function is using the DefAutoSelectWidth value initializes Title.Width for the new column.

See also: TXDBGridColumns.AddAutoSelect, TColumn.Width, DefAutoSelectTitle

DefAutoExpandTitle variable

Specifies the default title for AutoExpand column.

Unit
XDBGrids

```
const
  DefAutoExpandTitle: string = 'Expand'; {* ver. 6.6 *}
```

Description
When AddAutoExpandfunction is using the DefAutoExpandTitle value initializes Title.Caption for the new column.

See also: TXDBGridColumns.AddAutoExpand, TColumnTitle.Caption, DefAutoExpandWidth

DefAutoExpandWidth variable

Specifies the default width for AutoExpand column.

Unit
XDBGrids

```
const
  DefAutoExpandWidth: Integer = 48; {* ver. 6.6 *}
```

Description
When AddAutoExpand function is using the DefAutoExpandWidth value initializes Title.Width for the new column.

See also: TXDBGridColumns.AddAutoExpand, TColumn.Width, DefAutoExpandTitle

DefPickTextFormat variable

Specifies the default format for PickText list items.

Unit
XDBGrids

```
const
  DefPickTextFormat: string = '%1:s - %0:s'; {* ver. 4.3 *}
```

Description
When PickTextFormat property is empty, the DefPickTextFormat is using by FormatPickText function.

See also: TXColumn.PickText, TXColumn.PickTextFormat, TXColumn.FormatPickText

RegisterChangeOrder procedure

Registers typical DataSet class to automatically change order in the grid.

Unit
XDBGrids

```
const
  AscOnly = True;
  AscDesc = False;

type
  TChangeOrder = (coChangeIndexFields, coChangeOrderFields,
    coChangeSQLOrderFields, coCustomChangeOrder);
  TValidChangeOrder = function(DataSet: TDataSet; ChangeOrder: TChangeOrder):
    Boolean;

procedure RegisterChangeOrder(ClassName: string; ChangeOrder: TChangeOrder;
  AscendOnly: Boolean; IsValid: TValidChangeOrder = nil; PropName: string =
  ''; Suffix: string = '');
```

Description
Use RegisterChangeOrder procedure to register typical method of change sorting order in DataSet linked to the grid. TXDBGrid can automatically change sorting order for registered DataSet class when dgAutoUpdateOrder option is included in OptionsEx. When OrderFields (sorting markers) changes in the grid the ChangeDataSetOrder universal method is called. All standard DataSet descendats (except unidirectional) are registered by default to automatically change order (ADO, BDE, CDS, DBX, IBX). To register other typical DataSets you should call this method before.

ClassName parameter determines class name of registered DataSet. Each DataSet is registered exactly for the class (not for descendats). ChangeOrder parameter determines selected method of change sorting order available in registered DataSet. AscendOnly parameter determines that dgMarkerAscendOnly option should be included in Options property to use registered method without Beeps (use AscOnly/AscDesc constants). See also dgAutoAscendOnly option in OptionsEx.

You can also specify optionally IsValid function, if any other conditions must be meet to can use registered method for registered DataSet. This function will be called each time before using the registered method. You can also specify PropName parameter, if the name of published property in registered DataSet is different that expected.

`{* ver. 4.31 *}` You can also specify optionally Suffix parameter (for coChangeIndexFields only). When Suffix parameter is omitted (or empty) the default suffix `'ASC;DESC;'` will be used to retrieve and change current sorting order in data set. For other suffixes, you must explicity specify the ASC and DESC modifiers when they are accepted by DataSet. For example, the AnyDAC's data sets accept the modifiers `':A;:D;:N'`. Use semicolon to separate ASC and DESC modifiers. When none modifier is accepted by data set you must register the method with parameter AscOnly.

The following are possible values of ChangeOrder parameter:

Value	Meaning
coChangeIndexFields	Use ChangeIndexFields method to modify IndexFieldNames property in registered DataSet.
coChangeOrderFields	Use ChangeOrderFields method to modify CommandText property in registered DataSet.
coChangeSQLOrderFields	Use ChangeSQLOrderFields method to modify SQL property in registered DataSet.

coCustomChangeOrder	Do not use with RegisterChangeOrder procedure. Use RegisterCustomChangeOrder instead.

You can register more than one method for one ClassName (especially with different IsValid functions). Last method registered for the ClassName will be used as first (if IsValid returns True). It allows to select methods "on the fly" for any ClientDataSet depend on current conditions. For all pre-registered DataSets the methods with AscendOnly = False are prefered.

Please, open **XDBGridsADD.pas** unit to add RegisterChangeOrder procedure for additional typical dataset. This unit is fully useful and may be added to your application, if you are using these datasets.

See also: RegisterCustomChangeOrder, UnregisterChangeOrder

RegisterCustomChangeOrder procedure

Registers specific DataSet class to automatically change order in the grid.

Unit
XDBGrids

```
type
  TCustomChangeOrder = procedure(DBGrid: TXDBGrid; AOrderFields: WideString =
    '');
  TCustomCurrentOrder = function(DBGrid: TXDBGrid): WideString;

procedure RegisterCustomChangeOrder(ClassName: string; ChangeOrder:
  TCustomChangeOrder; CurrentOrder: TCustomCurrentOrder; AscendOnly: Boolean;
  IsValid: TValidChangeOrder = nil);
```

Description
Use RegisterCustomChangeOrder procedure to register specific method of change sorting order in DataSet linked to the grid. TXDBGrid can automatically change sorting order for registered DataSet class when dgAutoUpdateOrder option is included in OptionsEx. When OrderFields (sorting markers) changes in the grid the ChangeDataSetOrder universal method is called. All standard DataSet descendats (except unidirectional) are registered by default with using RegisterChangeOrder procedure to automatically change order (ADO, BDE, CDS, DBX, IBX). To register other specific DataSets you should call this method before.

ClassName parameter determines class name of registered DataSet. Each DataSet is registered exactly for the class (not for descendats). ChangeOrder parameter specifies custom procedure which can change sorting order in registered DataSet. CurrentOrder parameter specifies custom function which can retrieve current sorting order from registered DataSet. AscendOnly parameter determines that dgMarkerAscendOnly option should be included in Options property to use registered methods without Beeps. See also dgAutoAscendOnly option in OptionsEx.

You can also specify optionally IsValid function, if any other conditions must be meet to can use registered ChangeOrder method for registered DataSet. This function will be called each time before using ChangeOrder method. Registered CurrentOrder function must always return current sorting order.

You can register more than one method for one ClassName (especially with different IsValid functions). Last method registered for the ClassName will be used as first (if IsValid returns True). It allows to select methods "on the fly" for any ClientDataSet depend on current conditions.

Please, open **XDBGridsADO.pas** unit to see an example how to write complete RegisterCustomChangeOrder procedure. This unit is fully useful and may be added to your application, if you are using TADOQuery. It implements using in run-time Sort public property to automatically change sorting other in TADOQuery.

See also: RegisterChangeOrder, UnregisterChangeOrder

UnRegisterChangeOrder procedure

Unregisters method for DataSet class registered prior to automatically change order in the grid.

Unit
XDBGrids

```
procedure UnRegisterChangeOrder(ClassName: string; ChangeOrder: TChangeOrder);
```

Description
Use UnRegisterChangeOrder procedure to unregister method of change sorting order in DataSet linked to the grid. UnregisterChangeOrder procedure unregisters method of ChangeOrder type for ClassName registered prior by using RegisterChangeOrder or RegisterCustomChangeOrder procedure.

You may need to call this procedure if you don't want to modify CommandText property by ChangeOrderFields method and leave ChangeIndexFields (second) method to automatically change only IndexFieldNames property in any ClientDataSet.

See also: RegisterChangeOrder, RegisterCustomChangeOrder

AutoNumberStr function

Converts an AutoNumber value to a string.

Unit
XDBGrids

```
function AutoNumberStr(Value: Integer): string; {* ver. 4.3 *}
```

Description
AutoNumberStr converts an integer (AutoNumber value) into a string with using DefAutoNumberFormat containing the format of AutoNumber value representation. If Value is less or equal 0, the function returns empty string.

See also: DefAutoNumber, TXColumn.AutoNumber

GetTrueColor function

Converts clDefault/clNone color value to it's true equivalent.

Unit
XDBGrids

```
function GetTrueColor(Color, DefaultColor: TColor; NoneColor: TColor =
clDefault): TColor; {* ver. 5.1 *}
```

Description
GetTrueColor function returns Color value, if the value is different than clDefault and clNone. If Color value is clDefault the function returns DefaultColor parameter. If Color value is clNone, the function return NoneColor parameter (when was specified). This function is useful when you need to use true color value instead clDefault/clNone.

Using:

```
  with XDBGrid1 do
    ColorDialog1.Color := GetTrueColor(HighlightText, DefaultHighlightText,
Font.Color);
```


ScanStr function

Scans string to retrieve substrings separated by delimiter.

Unit
XDBGrids

```
const
  DefaultDelimiter: string = ',';

function ScanStr(var S: string; Delimiter: string = ''): string;
```

Description
Use ScanStr to retrieve (and return as Result) a left substring of S string separated by Delimiter. The right substring of S string (after Delimiter) is return as new S string. When Delimiter is empty the value of DefaultDelimiter global variable will be used as Delimiter. This function is useful to split several words separated by any delimiter to separate words.

For example after:

```
  S := 'one,two,three,four,five';
  while S <> '' do Memo1.Lines.Add(ScanStr(S));
```

The result in Memo1.Lines is:

```
  'one'
  'two'
  'three'
  'four'
  'five'
```

See also: ScanInt

ScanInt function

Scans string to retrieve numbers separated by delimiter.

Unit
XDBGrids

```
function ScanInt(var S: string; Default: Integer; Delimiter: string = ''):
Integer;
```

Description
Use ScanInt to retrieve (and return as Result) a left substring of S string separated by Delimiter. The right substring of S string (after Delimiter) is return as new S string. When left substring is not properly Integer value the Default parameter will be return as Result. When Delimiter is empty the value of DefaultDelimiter global variable will be used as Delimiter. This function is useful to split several numbers separated by any delimiter to separate numbers. ScanInt calls internally ScanStr function.

See also: ScanStr

CommaToSemicolon function

Returns a string with all occurrences of comma replaced by semicolon.

Unit
XDBGrids

```
function CommaToSemicolon(const S: string): string; overload;
function CommaToSemicolon(const S: WideString): WideString; overload; //
C++Builder/Delphi 2006, 2007 Win32 only
```

Description
CommaToSemicolon replaces all occurrences of comma with the semicolon.

See also: SemicolonToComma

SemicolonToComma function

Returns a string with all occurrences of semicolon replaced by comma.

Unit
XDBGrids

```
function SemicolonToComma(const S: string): string; overload;
function SemicolonToComma(const S: WideString): WideString; overload; //
C++Builder/Delphi 2006, 2007 Win32 only
```

Description
SemicolonToComma replaces all occurrences of semicolon with the comma.

See also: CommaToSemicolon

IndexFieldsToOrderFields function

Converts IndexFields string to OrderFields string.

Unit
XDBGrids

```
function IndexFieldsToOrderFields(IndexFields: string; Suffix: string = ''):
string; overload; {* ver. 5.0 *}
function IndexFieldsToOrderFields(IndexFields: WideString; Suffix: string =
''): WideString; overload; {* ver. 4.31 *} // C++Builder/Delphi 2006, 2007
Win32 only
```

Description
Call IndexFieldsToOrderFields to convert IndexFields string to OrderFields string with using Suffix modifier. See also CurrentIndexFields method.

See also: OrderFieldsToIndexFields

OrderFieldsToIndexFields function

Converts OrderFields string to IndexFields string.

Unit
XDBGrids

```
function OrderFieldsToIndexFields(OrderFields: string; Suffix: string = ''):
string; overload; {* ver. 5.0 *}
function OrderFieldsToIndexFields(OrderFields: WideString; Suffix: string =
''): WideString; overload; {* ver. 4.31 *} // C++Builder/Delphi 2006, 2007
Win32 only
```

Description
Call OrderFieldsToIndexFields to convert OrderFields string to IndexFields string with using Suffix modifier. See also ChangeIndexFields method.

See also: IndexFieldsToOrderFields

ExtractOrderFields function

Parses the SQL statement and returns fields followed by ORDER BY clause.

Unit
XDBGrids

```
function ExtractOrderFields(SQL: TStrings): string; overload;
function ExtractOrderFields(SQL: TWideStrings): WideString; overload; //
C++Builder/Delphi 2006 and higher
```

Description
Call ExtractOrderFields to retrieve fields followed by ORDER BY clause in SQL statement. ExtractOrderFields work correct when ORDER BY clause fill whole single line in the SQL statement or ORDER BY is the last clause in the line.

See also: ModifyOrderFields

ModifyOrderFields function

Parses the SQL statement and replaces fields followed by ORDER BY clause by OrderFields parameter.

Unit
XDBGrids

```
procedure ModifyOrderFields(SQL: TStrings; OrderFields: string); overload;
procedure ModifyOrderFields(SQL: TWideStrings; OrderFields: WideString);
overload; // C++Builder/Delphi 2006 and higher
```

Description
Call ModifyOrderFields to replace fields followed by ORDER BY clause in SQL statement by OrderFields value. ModifyOrderFields work correct when ORDER BY clause fill whole single line in the SQL statement or ORDER BY is the last clause in the line.

See also: ExtractOrderFields

SplitOrderFields function

Splits OrderFields to all and descending index fields.

Unit
XDBGrids

```
procedure SplitOrderFields(OrderFields: string; var Fields, DescFields:
string); overload; {* ver. 5.0 *} // C++Builder/Delphi 2006 and higher
procedure SplitOrderFields(OrderFields: WideString; var Fields, DescFields:
WideString); overload;
```

Description
Call SplitOrderFields to split one OrderFields string to Fields and DescFields index fields to store them into TIndexDef objects.

See also: UnifyOrderFields

UnifyOrderFields function

Unifies all and descending index fields into OrderFields.

Unit
XDBGrids

```
function UnifyOrderFields(Fields, DescFields: string; Descending: Boolean):
string; overload; {* ver. 5.0 *}
function UnifyOrderFields(Fields, DescFields: WideString; Descending:
Boolean): WideString; overload; // C++Builder/Delphi 2006, 2007 Win32 only
```

Description
Call UnifyOrderFields to unify Fields and DescFields index fields form TIndexDef objects into one OrderFields string. If DescFields string is empty and Descending is True, all Fields are unified as DescFields.

See also: SplitOrderFields

ValidFieldNames function

Unifies the list of field names depending on fields in DataSet.

Unit
XDBGrids

```
function ValidFieldNames(DataSet: TDataSet; FieldNames: string; SingleField:
Boolean = False): string; overload; {* ver. 5.0 *}
function ValidFieldNames(DataSet: TDataSet; FieldNames: WideString;
SingleField: Boolean = False): WideString; overload; {* ver. 5.0 *} //
C++Builder/Delphi 2006, 2007 Win32 only
```

Description
Call ValidFieldNames to unify FieldNames parameter depending on field names in DataSet The FieldNames parameter should contain the list of field names separated by semicolon. Set SingleField to True when function should return only one (first) name of field.

See also: SplitOrderFields

TVAlignment type

Specifies how text is vertical aligned within a cell.

Unit
XDBGrids

```
type
  TVAlignment = (tvTopJustify, tvBottomJustify, tvCenter);
```

Description
The following are possible values of TVAlignment:

Value	Meaning
tvTopJustify	Text is top-justified: Lines all begin at the top edge of the cell.
tvCenter	Text is centered in the cell.
tvBottomJustify	Text is right-justified: Lines all end at the bottom edge of the cell.

See also: TAlignment

TXDBColumn

TXDBColumn component represents a standalone column editor.

Unit
XDBEditor

Description
TXDBColumn component represents a standalone column editor. TXDBColumn performs the same functionality as inplace editor for the selected column in TXDBGrid component. TXDBColumn directly uses all column's properties set in TXDBGrid. The result of data editing in TXDBColumn is returned to the TXDBGrid. TXDBColumn functionality is implemented in TXDBDataEditor class.

See also: TXColumn, TXDBEditor

TXDBDataEditor

TXDBDataEditor class represents a common single-line editor.

Unit
XDBEditor

Description
TXDBDataEditor class represents a common single-line editor that can display and edit a field in a dataset or a column in TXDBGrid.

See also: TXDBColumn, TXDBEditor

TXDBDataEditor.Column

Specifies the TColumn object that is associated with the editor.

```
property Column: TXColumn; {* ver. 6.4 *}
```

Description
Read Column to get direct access to the contents and properties of the data-grid column. Use Column to change the contents of the data-grid column programmatically.

See also: TXDBDataEditor.DataColumn, TXDBDataEditor.Field

TXDBDataEditor.DataColumn

Specifies the column from which the editor displays data.

```
property DataColumn: string; {* ver. 6.4 *}
```

Description
Set DataColumn to the column name of the data-grid's column for the editor. Access by the editor to the dataset in which the column's field is located is provided by a TXDBGrid component, specified in the DBGrid property.

When DataColumn property is set, this property identifies also a field in DataSource component. When DataColumn is not explicit set, the DataField property identifies also the column in TXDBGrid component. You must set DataColumn property only when DBGrid contains more then one column with the same FieldName.

See also: TXDBDataEditor.Column, TXDBDataEditor.DataField, TXDBDataEditor.DBGrid

TXDBDataEditor.DataField

Specifies the field from which the editor displays data.

```
property DataField: string; {* ver. 6.4 *}
```

Description
Set DataField to the field name of the field component for the editor. Access by the editor to the dataset in which the field is located is provided by a TDataSource component, specified in the DataSource property.

When DBGrid property is set, the DataField property identifies also a column in TXDBGrid component. When DataColumn property was set first, you can't change DataField property unless the DataColumn property will be cleared.

See also: TXDBDataEditor.Field, TXDBDataEditor.DataColumn, TXDBDataEditor.DataSource

---

TXDBDataEditor.DataSource

Links the editor to the dataset that contains the field it represents.

```
property DataSource: TDataSource; {* ver. 6.4 *}
```

Description
Use DataSource to specify the data source component through which the data from a dataset component is provided to the editor. To represent the data for a field, both the DataSource and the DataField properties must be set.

When DBGrid property is set, the DataSource property is assigned automaticaly.

See also: TXDBDataEditor.DataField, TXDBDataEditor.DBGrid

TXDBDataEditor.DBGrid

Links the editor to the data-grid that contains the column it represents.

```
property DBGrid: TXDBGrid; {* ver. 6.4 *}
```

Description
Use DBGrid to specify the data-grid component through which the data from a dataset component is provided to the editor. To represent the data for a column, both the DBGrid and the DataColumn properties must be set.

When DBGrid property is set, the DataSource property is assigned automaticaly.

See also: TXDBDataEditor.DataColumn, TXDBDataEditor.DataSource

TXDBDataEditor.DropDownOnly

Specifies a functionality of drop-down list.

```
property DropDownOnly: Boolean; {* ver. 6.4 *}
```

Description
Set DropDownOnly to True, to create a drop-down list with no edit box. When DropDownOnly is True, the user cannot enter text manually.
Set DropDownOnly to False, to create a drop-down list with an edit box for manually entered text by the user.

The DropDownOnly property works effective for any kind of drop-down list (pick list, lookup list, calendar, calculator).

See also: TXDBCustomEditor.CalendarActive, TXDBCustomEditor.PickDataActive, TXDBCustomEditor.LookupActive

TXDBDataEditor.Field

Specifies the TField object for the database field the editor represents.

```
property Field: TField; {* ver. 6.4 *}
```

Description
Read Field to get direct access to the contents and properties of the database field. Use Field to change the contents of the database field programmatically.

See also: TXDBDataEditor.DataField, TXDBDataEditor.Column

TXDBDataEditor.Create

Creates and initializes a TXDBDataEditor instance.

```
constructor Create(AOwner: TComponent); override; {* ver. 6.4 *}
```

Description
Use Create to programmatically instantiate a data editor. TXDBColumn added in the form designer is created automatically.

AOwner is the component that is responsible for freeing the data editor instance. It becomes the value of the Owner property.

See also: TXDBDataEditor.Destroy

TXDBDataEditor.Destroy

Destroys the data editor object.

```
destructor Destroy; override; {* ver. 6.4 *}
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free checks that the data editor is not nil, and only then calls Destroy.

See also: TXDBDataEditor.Create

TXDBLabeledEditor

TXDBLabeledEditor class represents a common single-line editor with label.

Unit
XDBEditor

Description
TXDBLabeledEditor class represents a common single-line editor with label that can display and edit a field in a dataset or a column in TXDBGrid.

See also: TXDBColumn, TXDBEditor

TXDBLabeledEditor.EditorLabel

Specifies the control that implements the label.

```
property EditorLabel: TXDBEditorLabel; {* ver. 6.6 *}
```

Description
Use EditotLabel to work with the label that is associated with this labeled editor control. Use this label's properties to specify the caption that appears on the label, the position of the caption within the label, the font used to write that caption, and so on. At design time, expand the EditorLabel property in the Object Inspector to set its properties.

See also: TXDBLabeledEditor.LabelVisible, TXDBLabeledEditor.LabelPosition, TXDBLabeledEditor.LabelSpacing

TXDBLabeledEditor.LabelPosition

Specifies the position of the label relative to the editor control.

```
property LabelPosition: TLabelPosition; {* ver. 6.6 *}
```

Description
Set LabelPosition to indicate where the label sits relative to the editor control.

See also: TXDBLabeledEditor.EditorLabel, TXDBLabeledEditor.LabelVisible, TXDBLabeledEditor.LabelSpacing

TXDBLabeledEditor.LabelSpacing

Specifies the distance, in pixels, between the label and the editor region.

```
property LabelSpacing: Integer; {* ver. 6.6 *}
```

Description
Set LabelSpacing to indicate the distance, in pixels, between the label and the editor control.

See also: TXDBLabeledEditor.EditorLabel, TXDBLabeledEditor.LabelPosition, TXDBLabeledEditor.LabelVisible

TXDBLabeledEditor.LabelVisible

Specifies whether the label appears on screen.

```
property LabelVisible: Boolean; {* ver. 6.6 *}
```

Description
Use the LabelVisible property to control the visibility of the label. If LabelVisible is True, the label appears. If LabelVisible is False, the label is not visible.

See also: TXDBLabeledEditor.EditorLabel, TXDBLabeledEditor.LabelPosition, TXDBLabeledEditor.LabelSpacing

TXDBEditor

TXDBEditor component represents standalone field editor.

Unit
XDBEditor

Description
TXDBEditor component represents a standalone field editor. TXDBEditor performs the same functionality for a field in DataSet as inplace editor in TXDBGrid component. TXDBEditor allows to set properties similar to the column properties in TXDBGrid. The result of data editing in TXDBEditor is stored directly to the DataSet. TXDBEditor functionality is implemented in TXDBCustomEditor class.

When DataSource property is not assigned TXDBEditor can work as non db-aware editor.

See also: TXDBColumn, TXDBCustomEditor

TXDBCustomEditor

TXDBCustomEditor class represents a custom field editor.

Unit
XDBEditor

Description
TXDBCustomEditor class represents a custom field editor. TXDBCustomEditor allows to change the value of a field on the current record in a dataset either by selecting an item from a list or by typing in the edit box part of the control. TXDBCustomEditor can be customized to enable or disable typing in the edit region of the control, to display the list as a drop down list or as a permanently displayed list when DropDownOnly property is True.

When DataSource property is not assigned TXDBEditor can work as non db-aware editor.

See also: TXDBDataEditor, TXDBEditor

TXDBCustomEditor.ButtonStyle

Determines whether and how the user can select values for the editor from a list, menu or other kind of drop-down list.

```
type
  TEditorButtonStyle = (ebsAuto, ebsEllipsis, ebsNone, ebsDropDown,
    ebsDropDownMenu, ebsCalendar, ebsCalculator);

property ButtonStyle: TEditorButtonStyle; {* ver. 6.4 *}
```

Description
The ButtonStyle property determines how users can edit the data in the editor. These are the possible values of ButtonStyle:

Value	Meaning
ebsAuto	If the editor is associated with a lookup list or has a value assigned to its PickList property, the control displays a combo box in the editor. The user can choose a value from the drop-down list. If the editor is associated with TDateField or TDateTimeField the user can choose date value from the drop-down calendar. Time part of TDateTime value leaves unchanged. See also: ebsCalendar.
ebsEllipsis	The editor displays an ellipsis button that the user can click to choose a value.

	Clicking the ellipsis button triggers an OnEditButtonClick event.
ebsNone	No combo box or ellipsis button is provided. The user cannot select the editor's content from a list.
ebsDropDown	The editor displays a drop-down button that the user can click to choose a value. Clicking the drop-down button triggers an OnEditButtonClick event.
ebsDropDownMenu	The editor displays a drop-down button that the user can click to choose a value from drop-down menu. Clicking the drop-down button DropDownMenu is automatically dropped down.
ebsCalendar	The editor displays a drop-down button that the user can click to choose a date from drop-down calendar. Clicking the drop-down button the month calendar is automatically dropped down. The calendar requires date value compatible with ShortDateFormat global variable. You can also use ebsCalendar value when any text field contains date or datetime value.
ebsCalculator	The editor displays a drop-down button that the user can click to calculate input value in drop-down calculator. Clicking the drop-down button the comfortable calculator is automatically dropped down. The calculator is similar in use to Microsoft Calculator.

See also: TXColumn.ButtonStyle

TXDBCustomEditor.CalendarActive

Specifies whether the editor has a drop-down calendar.

```
property CalendarActive: Boolean; {* ver. 6.4 *}
```

Description
Use CalendarActive property to determine the editor has defined drop-down calendar.

CalendarActive is a read-only property.

See also: TXDBCustomEditor.ButtonStyle, TXDBCustomEditor.LookupActive, TXDBCustomEditor.PickDataActive

TXDBCustomEditor.CalendarDepth

Specifies maximum depth of drop-down calendar in the editor.

```
property CalendarDepth: TCalendarDepth; {* ver. 6.5 *}
```

Description
Use CalendarDepth property to select maximum depth of drop-down calendar in the editor.

See also: TXDBCustomEditor.ButtonStyle, TXDBCustomEditor.CalendarActive

TXDBCustomEditor.DelimiterChar

Indicates when multi selected values are delimited by specified character.

```
property DelimiterChar: Char; {* ver. 6.7 *}
```

Description
When the editor has associated MultiSelectList and property DelimiterLine is False, the multi selected values are delimited by character specified in DelimiterChar property.

See also: TXDBCustomEditor.DelimiterLine, TXDBCustomEditor.MultiSelectList

TXDBCustomEditor.DelimiterLine

Indicates when multi selected values are delimited by new line.

```
property DelimiterLine: Boolean; {* ver. 6.7 *}
```

Description
When MultiSelectListis associated with the editor and DelimiterLine property is True then multi selected values are delimited by new line (CR LF). When DelimiterLine is False, multi selected values are delimited by DelimiterChar.

See also: TXDBCustomEditor.DelimiterChar, TXDBCustomEditor.MultiSelectList

TXDBCustomEditor.DelimiterText

Indicates when multi selected values are delimited by few fixed characters.

```
property DelimiterText: string; {* ver. 8.0 *}
```

Description
When the editor has associated MultiSelectList and property DelimiterLine is False, the multi selected values can be delimited by few fixed characters specified in DelimiterText property. Remember, when DelimiterText is not empty the DelimiterChar is also used internally.

See also: TXDBCustomEditor.DelimiterChar, TXDBCustomEditor.MultiSelectList

TXDBCustomEditor.DrawingStyle

Specifies the drawing style of the drop-down lists.

```
property DrawingStyle: TGridDrawingStyle; {* ver. 6.5 *}
```

Description
Use DrawingStyle to select drawing style for drop-down lists. The possible values of DrawingStyle are the same as for property TXDBGrid.DrawingStyle.

See also: TXDBCustomEditor.ButtonStyle, TXCustomDBGrid.DrawingStyle

TXDBCustomEditor.DropDownMenu

Identifies the drop-down menu associated with the editor.

```
property DropDownMenu: TPopupMenu; {* ver. 6.4 *}
```

Description
Assign a value to DropDownMenu to make a drop-down menu appear when the user clicks the edit button in the editor. If the ButtonStyle property is ebsAuto when DropDownMenu is assigned, the ButtonStyle property is automatically changed to ebsDropDownMenu and the edit button will be appear in the editor. If the ButtonStyle is ebsDropDownMenu when DropDownMenu is removed, the ButtonStyle property is automatically changed to ebsAuto.

See also: TXDBCustomEditor.ButtonStyle

TXDBCustomEditor.DropDownRows

Specifies the number of lines displayed in the editor's drop-down list.

```
property DropDownRows: Integer; {* ver. 6.4 *}
```

Description
DropDownRows determines the number of lines of text displayed in the drop-down list associated with the editor. This property is used only if ButtonStyle is set to ebsAuto and the editor has a lookup list or pick list associated with it.

When DropDownRows property is 0 (default) and drop-down list has more then MaxDropDownRows items, then MaxDropDownRows items is displayed in the drop-down list, otherwise all list items are displayed.

See also: TXDBCustomEditor.DropDownWidth, MaxDropDownRows

TXDBCustomEditor.DropDownWidth

Specifies a width of the editor's drop-down list.

```
property DropDownWidth: Integer; {* ver. 6.4 *}
```

Description
DropDownWidth determines a width of the drop-down list associated with the editor. This property is used only if ButtonStyle is set to cbsAuto and the editor has a lookup list or pick list associated with it. If DropDownWidth value is less then editor's width the editor's width will be used as the width of the drop-down list.

When DropDownWidth property is 0 (default) and auto-calculated width of drop-down list is greater then MaxDropDownWidth, then displayed width of drop-down list is limited to MaxDropDownWidth.

See also: TXDBCustomEditor.DropDownRows, MaxDropDownWidth

TXDBCustomEditor.ImageMargin

Specifies width of the left margin in pixels to display image inside.

```
property ImageMargin: Integer; {* ver. 6.5 *}
```

Description
Use ImageMargin to determine width of the left margin for the editor box. When ImageMargin is greater than 0 and Images are defined, the image is drawn inside the margin area of editor box. When ImageMargin is 0 the image is not drawn inside editor.

See also: TXColumn.ImageMargin, TXDBCustomEditor.Images

TXDBCustomEditor.Images

Determines which image list is associated with the editor.

```
property Images: TImageList; {* ver. 6.5 *}
```

Description
Use Images to provide a customized list of bitmaps that can be displayed in the editor. The image selected from Images is displayed only when ImageMargin is greater then 0. The selected image is always centered in the margin area. Use OnCalcImageIndex event to change default image index. Images are also visible on dropdown list when loShowImages option is included in ListOptions property.

See also: TXColumn.Images, TXDBCustomEditor.ImageMargin

TXDBCustomEditor.ItemIndex

Specifies the index of the current value in Items list.

```
property ItemIndex: Integer; {* ver. 6.5 *}
```

Description
You can use ItemIndex to read index of the current value in Items list or to change current value visibled in the editor box. Property ItemIndex is identical with PickIndex property and it was introduced to compatibility with TComboBoxonly.

See also: TXDBCustomEditor.PickIndex, TXDBCustomEditor.Items

TXDBCustomEditor.Items

Lists values that the user can select for the editor.

```
property Items: TStrings; {* ver. 6.5 *}
```

Description
The Items property points to a TStrings object. If ButtonStyle is ebsAuto, these strings appear in the drop-down list associated with the editor. Property Items is identical with PickList property and it was introduced to compatibility with TComboBox only.

See also: TXDBCustomEditor.PickList, TXDBCustomEditor.ItemIndex

TXDBCustomEditor.KeyField

Indicates the key field in the dataset to match when doing a lookup.

```
property KeyField: string; {* ver. 6.4 *}
property KeyField: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
You can use LookupDataSet, KeyField, LookupKeyField, LookupResultField, LookupListFields to create a lookup list for a fkData field associated with the editor. If ButtonStyle is ebsAuto and LookupColumn is defined, the LookupListFields values from LookupDataSet appear in the drop-down list associated with the editor.

LookupKeyField is the field in the LookupDataSet which value must match the KeyField associated with editor. The field specified in LookupKeyField must be of the same type as the KeyField associated with editor, or the lookup list can't work.

Matching the value of the LookupKeyField in the LookupDataSet with the value of KeyField associated with the editor determines a specific record in the LookupDataSet when the lookup list is dropped down. When the lookup list is closed, the value of the LookupResultField in the indicated record is assigned to the field associated with the editor.

See also: TXDBCustomEditor.LookupDataSet, TXDBCustomEditor.LookupKeyField, TXDBCustomEditor.LookupResultField, TXDBCustomEditor.LookupListFields, TXDBDataEditor.DataField

TXDBCustomEditor.ListOptions

Specifies various pick list, lookup list and other kind of drop-down list properties of the editor.

```
property ListOptions: TXListOptions; {* ver. 6.4 *}
```

Description
Set ListOptions to include various pick list and lookup list properties for the editor. The possible values of editor's ListOptions are the same as for the column's ListOptions except of loAutoDropDown and loAutoCloseEditor that are not used for editor.

See also: TXColumn.PickOptions

TXDBCustomEditor.LookupActive

Specifies whether the editor has a lookup list.

```
property LookupActive: Boolean; {* ver. 6.4 *}
```

Description
Use LookupActive property to determine the editor has defined lookup list.

LookupActive is a read-only property.

See also: TXDBCustomEditor.LookupColumn, TXDBCustomEditor.CalendarActive, TXDBCustomEditor.PickDataActive

TXDBCustomEditor.LookupColumn

Determines the kind of the editor's lookup list.

```
property LookupColumn: TXLookupColumn; {* ver. 6.4 *}
```

Description
Use LookupColumn property to read the kind of the editor's lookup list. The possible values of editor's LookupColumn are the same as the column's LookupColumn.

See also: TXDBCustomEditor.LookupDataSet, TXDBCustomEditor.KeyField, TXDBCustomEditor.LookupKeyField, TXDBCustomEditor.LookupResultField, TXDBCustomEditor.LookupListFields, TXDBDataEditor.DataField

TXDBCustomEditor.LookupDataSet

Identifies the dataset used to look up values for fkData field associated with the editor.

```
property LookupDataSet: TDataSet; {* ver. 6.4 *}
```

Description
You can use LookupDataSet, KeyField, LookupKeyField, LookupResultField, LookupListFields to create a lookup list for a fkData field associated with the editor. If ButtonStyle is ebsAuto and LookupColumn is defined, the LookupListFields values from LookupDataSet appear in the drop-down list associated with the editor.

Use LookupDataSet to specify the dataset to use for looking up field values in a field (associated with the editor) with a FieldKind of fkData. TXDBCustomEditor.LookupDataSet for a FieldKind of fkData is similar to TField.LookupDataSet for a FieldKind of fkLookup.

When the lookup list is dropped down, the record in the LookupDataSet that contains the correct value is found by matching the LookupKeyField in the LookupDataSet with the current value of the KeyField associated with the column.

When the lookup list is dropped down and LookupDataSet is close, it's automatically opened and then closed, when the lookup list is closed. It allows to use always fresh data in the lookup list. When the lookup list is dropped down and LookupDataSet is early opened, no open/close action is performed. In this case the developer must decide when LookupDataSet should be refreshed.

See also: TXDBCustomEditor.KeyField, TXDBCustomEditor.LookupKeyField, TXDBCustomEditor.LookupResultField, TXDBCustomEditor.LookupListFields, TXDBDataEditor.DataField

TXDBCustomEditor.LookupKeyField

Identifies the field in the lookup dataset to match when doing a lookup.

```
property LookupKeyField: string; {* ver. 6.4 *}
property LookupKeyField: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
You can use LookupDataSet, KeyField, LookupKeyField, LookupResultField, LookupListFields to create a lookup list for a fkData field associated with the editor. If ButtonStyle is ebsAuto and LookupColumn is defined, the LookupListFields values from LookupDataSet appear in the drop-down list associated with the editor.

LookupKeyField is the field in the LookupDataSet which value must match the KeyField associated with editor. The field specified in LookupKeyField must be of the same type as the KeyField associated with editor, or the lookup list can't work.

Matching the value of the LookupKeyField in the LookupDataSet with the value of KeyField associated with the editor determines a specific record in the LookupDataSet when the lookup list is dropped down.

When the lookup list is closed, the value of the LookupResultField in the indicated record is assigned to the field associated with the editor.

See also: TXDBCustomEditor.LookupDataSet, TXDBCustomEditor.KeyField, TXDBCustomEditor.LookupResultField, TXDBCustomEditor.LookupListFields, TXDBDataEditor.DataField

TXDBCustomEditor.LookupListFields

Identifies the field or fields whose values are displayed in the lookup list.

```
property LookupListFields: string; {* ver. 6.4 *}
property LookupListFields: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
You can use LookupDataSet, KeyField, LookupKeyField,, LookupResultField, LookupListFields to create a lookup list for a fkData field associated with the editor. If ButtonStyle is ebsAuto and LookupColumn is defined, the LookupListFields values from LookupDataSet appear in the drop-down list associated with the editor.

LookupListFields is the name of the field or fields in the LookupDataSet that are displayed in the drop-down list. LookupListFields can represent more than one field.

Before specifying LookupListFields, specify the LookupResultField property. If LookupListFields is not set, the lookup list displays LookupResultField field value by default. TXDBCustomEditor.LookupListFields is similar to TDBLookupControl.ListField.

You can also use LookupListFields property when the field associated with the editor is a fkLookup field. Then, you can specify more than one field (instead of Field.LookupResultField) displayed on the drop-down list. For a fkLookup field associated with the editor the LookupListFields should be specified from Field.LookupDataSet.

See also: TXDBCustomEditor.LookupDataSet, TXDBCustomEditor.KeyField, TXDBCustomEditor.LookupResultField, TXDBCustomEditor.LookupKeyField, TXDBDataEditor.DataField

TXDBCustomEditor.LookupListTitles

Specifies the titles for lookup list columns.

```
property LookupListTitles: string; {* ver. 6.4 *}
```

Description
You can show titles for lookup list columns by including loShowLookupListTitles option in ListOptions. The LookupListTitles property specifies titles (separated by semicolor) for each column (field) specified in LookupListFields. When this property is empty the field name is showing as title for each lookup list column.

See also: TXDBCustomEditor.LookupDataSet, TXDBCustomEditor.LookupKeyField, TXDBCustomEditor.LookupListFields, TXDBCustomEditor.LookupListWidths

TXDBCustomEditor.LookupListWidths

Specifies the widths for lookup list columns.

```
property LookupListWidths: string; {* ver. 6.4 *}
```

Description
You can show titles for lookup list columns by including loShowLookupListTitles option in ListOptions. The LookupListWidths property specifies widths (separated by semicolor) for each column (field) specified in LookupListFields. When this property is empty the default width is setting for each lookup list column.

See also: TXDBCustomEditor.LookupDataSet, TXDBCustomEditor.LookupKeyField, TXDBCustomEditor.LookupListFields, TXDBCustomEditor.LookupListTitles

TXDBCustomEditor.LookupParentField

Indicates the parent field for tree view in the lookup dataset.

```
property LookupParentField: string; {* ver. 7.0 *}
property LookupParentField: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
LookupParentField contains key value of parent row for tree view on the lookup list. LookupParentField is the field in the LookupDataSet which value must match the LookupKeyField. The field specified in LookupParentField must be of the same type as LookupKeyField, or the tree view in the lookup list can't work.

See also: TXDBCustomEditor.LookupDataSet, TXDBCustomEditor.LookupKeyField, TXDBCustomEditor.LookupTreeViewFields

TXDBCustomEditor.LookupResultField

Indicates the result field in the lookup dataset.

```
property LookupResultField: string; {* ver. 6.4 *}
property LookupResultField: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description
You can use LookupDataSet, KeyField, LookupKeyField,, LookupResultField, LookupListFields to create a lookup list for a fkData field associated with the editor. If ButtonStyle is ebsAuto and LookupColumn is defined, the LookupListFields values from LookupDataSet appear in the drop-down list associated with the editor.

LookupResultField is the field in the LookupDataSet which value must match the field associated with editor. The field specified in LookupResultField must be of the same type as the field associated with editor, or the lookup list can't work. These rules are not applied when LookupColumn is lcLookupValueField.

Matching the value of the LookupKeyField in the LookupDataSet with the value of KeyField associated with the editor determines a specific record in the LookupDataSet when the lookup list is dropped down. When the lookup list is closed, the value of the LookupResultField in the indicated record is assigned to the field associated with the editor.

See also: TXDBCustomEditor.LookupDataSet, TXDBCustomEditor.KeyField, TXDBCustomEditor.LookupKeyField, TXDBCustomEditor.LookupListFields, TXDBDataEditor.DataField

TXDBCustomEditor.LookupTreeViewField

Indicates the field for tree view in the lookup dataset.

```
property LookupTreeViewField: string; {* ver. 7.0 *}
property LookupTreeViewField: WideString; // Delphi 2006, 2007 Win32, 2009, 2010, XE only
```

Description

LookupTreeViewField indicates field in LookupDataSet for the tree view on the lookup list. LookupTreeViewField is one of the field on list LookupListFields.

See also: TXDBCustomEditor.LookupDataSet, TXDBCustomEditor.LookupParentField, TXDBCustomEditor.LookupListFields

TXDBCustomEditor.MultiSelectList

Indicates whether the editor has associated multi select list.

```
property MultiSelectList: Boolean; {* ver. 6.7 *}
```

Description

MultiSelectList property indicates wherher the editor has associated multi select list. MultiSelectList returs True only when loMultiSelectList option is included in ListOptions and DataType of editor's field is string or memo and the editor has associeted PickList/PickText or LookupColumn is lcLookupKeyField. When MultiSelectList is True the associated list allows to multi select values. Values selected on the list are stored to the field separated by DelimiterLine or DelimiterChar. Count of selected values are limited by maximum size of the field.

MultiSelectList is a readonly property.

See also: TXDBCustomEditor.ListOptions, TXDBCustomEditor.MultiSelectText

TXDBCustomEditor.MultiSelectText

Specifies how to display multi selected values in the editor.

```
type
  TMultiSelectText = (mstDefault, mstEllipsis);
property MultiSelectText: TMultiSelectText; {* ver. 6.7 *}
```

Description

When MultiSelectList is associated with the editor, MultiSelectText property specifies how to display multi selected values. These are the possible values of MultiSelectText property:

Value	Meaning
mstDefault	Multi selected values are displayed exactly as they are stored in the field (separated by DelimiterLine or DelimiterChar).
mstEllipsis	Only first selected value is displayed and it is finished by ellipsis when more then one value was selected on the list.

See also: TXDBCustomEditor.MultiSelectList, TXDBCustomEditor.ListOptions

TXDBCustomEditor.PickData

Determines a pick list currently used to show values for the editor.

```
property PickData: TStrings; {* ver. 6.4 *}
```

Description
The PickData property points to PickList or either PickText or either formatted PickText list defined for the editor depending on PickOptions and PickTextFormat properties. The PickData list always contains values currently visible in the editor box.

PickData is a read-only property.

See also: TXDBCustomEditor.PickDataList, TXDBCustomEditor.PickList, TXDBCustomEditor.PickText, TXDBCustomEditor.PickTextFormat, TXDBCustomEditor.PickOptions

TXDBCustomEditor.PickDataActive

Specifies whether the editor has a pick data list.

```
property PickDataActive: Boolean; {* ver. 6.4 *}
```

Description
Use PickDataActive property to determine the editor has defined a pick data list.

PickDataActive is a read-only property.

See also: TXDBCustomEditor.PickDataList, TXDBCustomEditor.LookupActive, TXDBCustomEditor.CalendarActive

TXDBCustomEditor.PickDataList

Determines a pick list visibled as drop-down list for the editor.

```
property PickDataList: TStrings; {* ver. 6.4 *}
```

Description
The PickDataList property points to PickList or either PickText or either formatted PickText list defined for the column depending on PickOptions and PickTextFormat properties. The PickDataList always contains values visibled on drop-down list.

PickDataList is a read-only property.

See also: TXDBCustomEditor.PickData, TXDBCustomEditor.PickList, TXDBCustomEditor.PickText, TXDBCustomEditor.PickTextFormat, TXDBCustomEditor.PickOptions

TXDBCustomEditor.PickIndex

Specifies the index of the current value in the pick data list.

```
property PickIndex: Integer; {* ver. 6.4 *}
```

Description
When PickDataActive property is True, you can use PickIndex to read index of the current value in the pick data list or change current value visibled in the editor box.

See also: TXDBCustomEditor.PickData, TXDBCustomEditor.PickDataActive

TXDBCustomEditor.PickList

Lists values that the user can select for the editor.

```
property PickList: TStrings; {* ver. 6.4 *}
```

Description
The PickList property points to a TStrings object. If ButtonStyle is ebsAuto, these strings appear in the drop-down list associated with the editor.

See also: TXDBCustomEditor.LoadPickList, TXDBCustomEditor.PickText, TXDBCustomEditor.ListOptions, TXDBCustomEditor.PickOptions

TXDBCustomEditor.PickOptions

Specifies various PickList and PickText properties of the editor.

```
property PickOptions: TXPickOptions; {* ver. 6.4 *}
```

Description
Set PickOptions to include various PickList and PickText properties for the editor. The possible values of editor's PickOptions are the same as for the column's PickOptions except of poAutoLoadList that is not supported by editor. Instead of using poAutoLoadList option you can perform LoadPickList method.

See also: TXColumn.PickOptions, TXDBCustomEditor.PickList, TXDBCustomEditor.PickText, TXDBCustomEditor.ListOptions

TXDBCustomEditor.PickText

Determines text for corresponding PickList's values that the user can select for the editor.

```
property PickText: TStrings; {* ver. 6.4 *}
```

Description
The PickText property points to a TStrings object. If poSelectPickText option is included in PickOptions and PickText is not empty, values from PickText are displayed instead of corresponding values from PickList (stored in dataset). If ButtonStyle is ebsAuto, these strings appear on the drop-down list associated with the editor.

See also: TXDBCustomEditor.LoadPickText, TXDBCustomEditor.PickList, TXDBCustomEditor.ListOptions, TXDBCustomEditor.PickOptions

TXDBCustomEditor.PickTextFormat

Specifies format string for PickText list formatting.

```
property PickTextFormat: string; {* ver. 6.4 *}
```

Description
Use PickTextFormat property to determine format string for PickText items. When poFormatPickText option is included in PickOptions the PickText items are formatted by using FormatPickText function.

If the string specified by the PickTextFormat property is empty, the PickText item value is formatted according to DefPickTextFormat variable.

When PickTextFormat is not empty it must contain format string (any text) with one, two or three argument specifiers. Each argument specifier must be appropriate to expected type according to SysUtils.Format function for strings and numeric values in order: **string**, **string**, **Integer**. The result of FormatPickText function is performed as:

```
Result := Format(PickTextFormat, [PickTextItem, PickListItem, ListIndex];
```


Use property editor to select one of most popular format for PickTextFormat. These are the most popular formats:

Format	Result
%0:s - %1:s	Kansas - KA
%0:s (%1:s)	Kansas (KA)
%0:s (%2:d)	Kansas (18)
%1:s - %0:s	KA - Kansas
%1:s (%0:s)	KA (Kansas)
%2:2d. %0:s	18. Kansas
%2:2d. %0:s (%1:s)	18. Kansas (KA)
%2:2d. %1:s (%0:s)	18. KA (Kansas)

See also: TXDBCustomEditor.PickText, TXDBCustomEditor.FormatPickText, TXDBCustomEditor.PickOptions, DefPickTextFormat

TXDBCustomEditor.SuppressRTL

Suppress the bi-directional mode for the editor.

```
property SuppressRTL: Boolean; {* ver. 6.4 *}
```

Description

When BiDiMode is set to bdRightToLeft this property allow to suppress right to left reading for the editor.

See also: TControl.BiDiMode, TControl.UseRightToLeftAlignment

TXDBCustomEditor.Create

Creates and initializes a TXDBCustomEditor instance.

```
constructor Create(AOwner: TComponent); override; {* ver. 6.4 *}
```

Description

Use Create to programmatically instantiate a custom editor. TXDBEditor added in the form designer is created automatically.

AOwner is the component that is responsible for freeing the data editor instance. It becomes the value of the Owner property.

See also: TXDBCustomEditor.Destroy

TXDBCustomEditor.Destroy

Destroys the custom editor object.

```
destructor Destroy; override; {* ver. 6.4 *}
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free checks that the custom editor is not nil, and only then calls Destroy.

See also: TXDBCustomEditor.Create

TXDBCustomEditor.FormatPickText

Formats a PickText item value.

```
procedure FormatPickText(ListIndex: Integer): string; {* ver. 6.4 *}
```

Description
FormatPickText formats the PickText item value given by ListIndex using the format given by PickTextFormat property.

If the string specified by the PickTextFormat property is empty, the PickText item value is formatted according to DefPickTextFormat variable.

When PickTextFormat is not empty it must contain format string (any text) with one, two or three argument specifiers. The each argument specifier must be appropriate to expected type according to SysUtils.Format function for strings and numeric values in order: **string**, **string**, **Integer**. The result of FormatPickText function is performed as:

```
  Result := Format(PickTextFormat, [PickTextItem, PickListItem, ListIndex];
```

See also: TXDBCustomEditor.PickText, TXDBCustomEditor.PickTextFormat, DefPickTextFormat

TXDBCustomEditor.LoadPickList

Loads unique items from dataset to the PickList.

```
procedure LoadPickList(PickListField: TField = nil; Sorted: Boolean = True);
{* ver. 6.4 *}
```

Description
Use LoadPickList method to load all unique values from PickListField's dataset to the PickList object. When PickListField parameter is nil, the Field linked to the editor will be use to load data. You can specify PickListField from any dataset (not only from dataset linked to the editor). When PickListField's dataset is not active, the dataset will be open and close after that.

When Sorted parameter is True, the PickList items are sorted in alphabetical order. When Sorted is False, the PickList items are loaded according to current order in dataset.

Note. When poAppendToList option is included in PickOptions, the LoadPickList appends only new values to the PickList.

See also: TXDBCustomEditor.PickList, TXDBCustomEditor.LoadPickText

TXDBCustomEditor.LoadPickText

Loads unique items pair from dataset to the PickList and PickText.

```
procedure LoadPickText(PickListField, PickTextField: TField; Sorted: Boolean =
True); {* ver. 6.4 *}
```

Description
Use LoadPickText method to load all unique values pair from PickListField/PickTextField's dataset to the PickList and PickText object. When PickListField or PickTextField parameter is nil, the PickList and PickText properties are cleared. You can specify PickListField/PickTextField from any dataset (not only from dataset linked to the editor), but both fields must came from the same dataset. When dataset is not active, the dataset will be open and close after that.

When Sorted parameter is True, the PickList/PickText items are sorted in alphabetical order. When poSortedByText option is included in PickOptionsthe alphabetical order depend on PickText items, otherwise the alphabetical order depend on PickList items. When Sorted is False, the PickList/PickText items are loaded according to current order in dataset.

Note. When poAppendToList option is included in PickOptions, the LoadPickText appends only new values pair to the PickList/PickText.

See also: TXDBCustomEditor.PickText, TXDBCustomEditor.LoadPickList

TXDBCustomEditor.SwitchListOptions

Include or exclude ListOptions depending on State.

```
procedure SwitchListOptions(Part: TXListOptions; State: Boolean); {* ver. 6.4
*}
```

Description
Use SwitchListOptions to include or exclude the Part of ListOptions property. When State is True the Part options are included, otherwise excluded.

See also: TXDBCustomEditor.SwitchPickOptions

TXDBCustomEditor.SwitchPickOptions

Include or exclude PickOptions depending on State.

```
procedure SwitchPickOptions(Part: TXPickOptions; State: Boolean); {* ver. 6.4
*}
```

Description
Use SwitchPickOptions to include or exclude the Part of PickOptions property. When State is True the Part options are included, otherwise excluded.

See also: TXDBCustomEditor.SwitchListOptions

TXDBCustomEditor.OnCalcBoldDays

Occurs whenever a new month is displayed in the drop-down calendar.

```
property OnCalcBoldDays: TCalcBoldDaysEvent; {* ver. 6.4 *}
```

Description
Use the OnCalcBoldDays event to initialize the display properties of a month. In particular, OnCalcBoldDays allows applications to bold individual days in the calendar (such as holidays).

Default value of BoldDay is False (for every Date). To specify that specific dates are to be bolded, set BoldDay to True. To specify that all sundays are to be bolded you can use: BoldDay := DayOfWeek(Date) = 1;

Notice. Be careful, OnCalcBoldDays event is fired for every showing day in calendar.

See also: TXCustomDBGrid.OnCalcBoldDays, TXDBCustomEditor.ButtonStyle, TXDBCustomEditor.ListOptions

TXDBCustomEditor.OnCalcImageIndex

Occurs when the editor needs an index to draw bitmap from Images.

```
property OnCalcImageIndex: TCalcImageIndexEvent; {* ver. 6.5 *}
```

Description
Write an OnCalcImageIndex event handler to change default index for the Images property. OnCalcImageIndex event is fired when ImageMargin is greater then 0 and editor has assigned Images property. Default value for Index is depend on field's value.

When the event is fired for editor (Sender is TXDBEditor object), the Column parameter is nil. When the event is fired for editor's dropdown list (Sender is TXDBPopupDataList object), the Column parameter points to the list's column with Images. When Sender is TXDBPopupDataList then TXDBPopupDataList(Sender).ParentEditor points to the editor object.

Images are visible on dropdown list only when loShowImages option is included in ListOptions property.

See also: TXCustomDBGrid.OnCalcImageIndex, TXDBCustomEditor.ImageMargin, TXDBCustomEditor.Images

TXDBCustomEditor.OnEditButtonClick

Occurs when the user clicks edit button in the editor.

```
property OnEditButtonClick: TNotifyEvent; {* ver. 6.4 *}
```

Description
Write an OnEditButtonClick event handler to take specific action when the user clicks edit button in the editor.

See also: TXDBCustomEditor.ButtonStyle

TXDBCustomEditor.OnGetText

Occurs when the grid retrieves Text from dataset.

```
type
  TEditorGetTextEvent = procedure(Sender: TObject; var Text: string;
    DisplayText: Boolean) of object;
property OnGetText: TEditorGetTextEvent; {* ver. 6.4 *}
```

Description
Write an OnGetText event handler to change Text retrieved from dataset.

See also: TXDBCustomEditor.OnSetText

TXDBCustomEditor.OnListCloseUp

Occurs when the editor's drop-down list has been closed.

```
type
  TEditorListCloseUpEvent = procedure(Sender, List: TObject; var Value:
    Variant; var Accept: Boolean) of object;
property OnListCloseUp: TEditorListCloseUpEvent ; {* ver. 6.4 *}
```

Description
Write an OnListCloseUp event handler to take specific action when the editor's list has been closed. The Sender parameter indicates the editor for the list. The List parameter holds the editor's list. Look at OnListDropDown to read, how to use List.

The Value parameter contains the value selected by the user. The Accept parameter contains the user's choice result. When Accept is True, the new Value was accepted by the user, when Accept is False, the user cancelled the choice. You can change the user's choice by modifying the Value and/or the Accept parameter.

See also: TXDBCustomEditor.OnListDropDown

TXDBCustomEditor.OnListDropDown

Occurs when the editor's drop-down list is dropped-down.

```
type
  TEditorListDropDownEvent = procedure(Sender, List: TObject) of object;
property OnListDropDown: TEditorListDropDownEvent; {* ver. 6.4 *}
```

Description
Write an OnListDropDown event handler to take specific action when the editor's list is dropped-down. OnListDropDown is triggered when the editor's list is ready to appear on the screen, but it's just not visible. The Sender parameter indicates the editor for the list. The List parameter holds the editor's list object. The type of List depends on ButtonStyle value for the editor.

Each drop-down list is derived from TXDBGridDropDownList class and declared in XDBDataList unit: TXDBEditorDataList, TXDBEditorPickList, TXDBEditorCalculator, TXDBEditorCalendar. If you want to modify the List object, you are doing it on your own risk.

See also: TXDBCustomEditor.OnListCloseUp

TXDBCustomEditor.OnSetText

Occurs when the grid stores Text to dataset.

```
type
  TEditorSetTextEvent = procedure(Sender: TObject; var Text: string) of
    object;
property OnSetText: TEditorSetTextEvent; {* ver. 6.4 *}
```

Description

Write an OnSetText event handler to change Text stored to dataset.

_____

See also: TXDBCustomEditor.OnGetText

TXQRGrid

TXQRGrid can preview and print records basis upon TXDBGrid component's settings.

Unit
XQRGrids

Description
TXQRGrid is a complementary component for TXDBGrid designed for dynamic report creating on the grounds of TXDBGrid current settings. TXQRGrid can print and preview content of TXDBGrid. TXQRGrid create in run-time complete report in QuickReport format. Report may be created in few parts (vertical bands) depend on selected paper size settings.

See also: TXDBGrid

TXQRGrid.AutoRowHeight

Determines automatic row height adjustment for all/selected columns.

```
type
  TXReportHeight = (rhDisabled, rhPartial, rhComplete);
property AutoRowHeight: TReportHeight;
```

Description
Set AutoRowHeight to rhPartial or rhComplete to cause automatic row height adjustment. When AutoRowHeight is rhComplete all columns take part in row height adjustment calculations. When AutoRowHeight is rhPartial only columns marked by AutoHeight take part in row height adjustment calculations. See also: roMultiPartRowHeight option.

Set AutoRowHeight to rhDisabled to create report with using fixed row height determined by LinesCount.

See also: TXColumnReport.AutoHeight

TXQRGrid.CustomCount

Indicates the number of records selected by developer.

```
property CustomCount: Integer;
```

Description
Use CustomCount to determine the number of records selected by developer. The value of CustomCount property is printed when you select rdCustomCount or rdCustomNoPerCount SysData. The CustomCount is designed to count records selected by developer. To calculate value of CustomCount the report must be prepared. When Prepare is finished the CustomCount property is initialized by the last value of CustomNumber. See also PrepareNeeded.

See also: TXQRGrid.CustomNumber, TXQRGrid.RecordCount, TXReportText.SysData

TXQRGrid.CustomNumber

Indicates the ordinal position of record calculated by developer.

```
property CustomNumber: Integer;
```

Description
Use CustomNumber to determine the ordinal position of record calculated by developer. The value of CustomNumber property is printed when you select rdCustomNo or rdCustomNoPerCount SysData. The value of CustomNumber should be incremented by developer for selected records. The CustomNumber is usually incremented in BeforeBandPrint event handler when PrintBand is True and Sender.BandType is rbDetail. The CustomNumber is initialized by "0" each time before BeforePrint event occurs.

See also: TXQRGrid.CustomCount, TXQRGrid.RecordNumber, TXReportText.SysData

TXQRGrid.FixedCols

Specifies the number of columns on the left of the grid that are repeated on each vertical band.

```
property FixedCols: Integer;
```

Description
Use FixedCols to specify, how many columns on the left of the grid should be repeated on each vertical band when multi-part report is created. Set FixedCols to -1 to assume current TXDBGrid.FixedCols value.

Note: Report must include at least one non-fixed column. Do not set FixedCols to a value greater than Columns.Count - 1. Total width of all fixed columns can not be greater then report's page width (without margins).

See also: TXQRGrid.FixedColsCount

TXQRGrid.FixedColsCount

Specifies the current number of columns on the left of the grid that are repeated on each vertical band.

```
property FixedColsCount: Integer;
```

Description
Use FixedColsCount to read the current number of columns that are repeated on each vertical band. FixedColsCount always returns value from range 0 to Columns.Count - 1. See also: FixedCols.

FixedColsCount is a read-only property.

See also: TXQRGrid.FixedCols

TXQRGrid.Options

Specifies various display and behavioral properties of the XQRGrid.

```
type
  TXReportOption = (roFirstPageHeader, roLastPageFooter, roCompression,
    roPrintIfEmpty, roShowProgress, roForceTrueTypes, roAllowFixedColor,
    roModalMode, roAllPagesTitle, roAllPagesLegend, roGroupForceNewPage,
    roSaveDfmAsText, roAllowVAlignment, roMultiPartAutoHeight,
    roKeepPrinterSetup, roKeepDesignPrinterSetup, roPreparePrinterSetup,
    roForceRGBColor, roAutoCalcPageWidth, roPrintHiddenColumns);
  TXReportOptions = set of TXReportOption;
property Options: TXReportOptions;
```

Description
Set Options to include the desired properties for the XQRGrid component. Options is a set drawn from the following values:

Value	Meaning
roFirstPageHeader	Page header will be printed on the first page.
roLastPageFooter	Page footer will be printed on the last page.
roCompression	Report will be compressed in memory during generation.
roPrintIfEmpty	Page will be also generated if there are no records in the dataset.
roShowProgress	Progress form will be displayed during printing of the report.
roForceTrueTypes	Bitmap fonts will be automatically replaced by true type fonts.
roAllowFixedColor	The FixedColor will be used for first FixedCols columns.
roModalMode	All other windows will be disabled during Preview.
roAllPagesTitle	Page title will be printed on all pages.
roAllPagesLegend	Page legend will be printed on all pages.
roGroupForceNewPage	Each group will be printed on new page (new column).
roSaveDfmAsText	SaveReport creates *.dfm file as text file.
roAllowVAlignment	Apply vertical text alignment in cells.
roMultiPartAutoHeight	Apply AutoRowHeight jointly by all vertical bands.
roKeepPrinterSetup	Save changes made in PrinterSetup dialog in run-time.
roKeepDesignPrinterSetup	Save changes made in PrinterSetup dialog in design-time.
roPreparePrinterSetup	Prepare report before calling PrinterSetup dialog.
roForceRGBColor	Replace system colors to RGB colors.
roAutoCalcPageWidth	Always fit PrinterPage.Width to the width of ReportDBGrid.
roPrintHiddenColumns	Print columns invisible in ReportDBGrid, when Report.Visible = True.

See also: TQuickReportOptions, TCustomQuickRep.Options, TCustomQuickRep.PrintIfEmpty, TCustomQuickRep.ShowProgress

TXQRGrid.PageCount

Indicates the number of printed pages.

```
property PageCount: Integer;
```

Description
Use PageCount to determine the number of printed pages. The value of PageCount property is printed when you select rdPageCount or rdPageNoPerCount SysData. To calculate value of PageCount property the report must be prepared. See also PrepareNeeded.

See also: TXReportText.SysData

TXQRGrid.PreparedNeeded

Determines when the report should be prepared before printing or preview.

```
property PrepareNeeded: TXReportVisible;
```

Description
Set PrepareNeeded to rvTrue to always prepare report before printing or preview. Set PrepareNeeded to rvFalse to never prepare report before printing or preview. Default value for PrapareNeeded is rvAuto. It means, the report will be prapared only when Prepare is needed to determine value of counters used in SysData.

When Prepare is finished the OnAfterPrepare event occurs. In this time all counters are just calculated.

See also: TXQRGrid.IsPrepareNeeded, TXQRGrid.AfterPrepare

TXQRGrid.PrieviewSettings

Represents the preview form specific options.

```
property PreviewSettings: TXQRPreviewSettings;
```

Description
Use the PreviewSettings property to set specific options for preview form. The PreviewSettings possibilities are accessible starting from QuickReport 3.6 Professional. For QuickReport 3.x Standard the PreviewSettings properties are of no importance. Some properties can be accesible starting from QuickReport 4.0 Professional.

See also: TXQRPreviewSettings

TXQRGrid.PrintCurrentRow

Determines printing report for current record only.

```
property PrintCurrentRow: Boolean;
```

Description
Set PrintCurrentRow to True to determine printing report only for the record marked as current in the ReportDBGrid component. See also PrintSelectedRows property.

Set PrintCurrentRow and PrintSelectedRows to False to determine printing report for all records that are shown in the ReportDBGrid component.

See also: TXQRGrid.PrintSelectedRows

TXQRGrid.PrinterPage

Represents the page layout for the report.

```
property PrinterPage: TQRPage;
```

Description
Use the PrinterPage property to set page layout information for the report. See TQRPage for more information.

Notice.
To assign value to PrinterPage.PaperSize in run-time you must add QRPrntr unit to the uses clause. Then, you can write: PrinterPage.PaperSize := A4;

To assign value to PrinterPage.Orientation in run-time you must add Printers unit to the uses clause. Then, you can write: PrinterPage.PaperSize := poLandscape;

See also: TQRPage, TCustomQuickRep.Page

TXQRGrid.PrinterSettings

Represents the printer specific options.

```
type
  TXQRPrinterSettings = class(TQuickRepPrinterSettings);
property PrinterSettings: TXQRPrinterSettings;
```

Description
Use the PrinterSettings property to set printer specific options. See also TQRPrinterSettingsfor more infomation. The PrinterSettings properties are dependent on installed version of QuickReport. Some properties can be accesible starting from QuickReport 3.6 or 4.0 Professional.

Notice. To assign value to some PrinterSettings properties in run-time you must add QRPrntr unit to the uses clause. Then, you can write e.g.: PrinterSettings.OutputBin := Manual;

See also: TQuickRepPrinterSettings, TCustomQuickRep.PrinterSettings

TXQRGrid.PrintSelectedRows

Determines printing report for selected records only.

```
property PrintSelectedRows: Boolean;
```

Description
Set PrintSelectedRows to True to determine printing report only for records marked as SelectedRows in the ReportDBGrid component. See also PrintCurrentRow property.

Set PrintCurrentRow and PrintSelectedRows to False to determine printing report for all records that are shown in the ReportDBGrid component.

See also: TXQRGrid.PrintCurrentRow

TXQRGrid.RecordCount

Indicates the number of printed records.

```
property RecordCount: Integer;
```

Description
Use RecordCount to determine the number of printed records. The value of RecordCount property is printed when you select rdRecordCount or rdRecordNoPerCount SysData. The RecordCount count only records that are really printed. To calculate value of RecordCount the report must be prepared. See also PrepareNeeded.

See also: TXQRGrid.RecordNumber, TXQRGrid.CustomCount, TXReportText.SysData

TXQRGrid.RecordNumber

Indicates the ordinal position of printed record.

```
property RecordNumber: Integer;
```

Description
Use RecordNumber to determine the ordinal position of printed record. The value of RecordNumber property is printed when you select rdRecordNo or rdRecordNoPerCount SysData or either when a column has property AutoNumber = True. The value of RecordNumber is incremented only for records that are really printed.

See also: TXQRGrid.RecordCount, TXQRGrid.CustomNumber, TXReportText.SysData

TXQRGrid.ReportAlign

Determines how the report is aligned within the page.

```
type
  TXReportAlign = (raLeft, raRight, raCenter, raStretch, raStretchAll);
property ReportAlign: TXReportAlign;
```

Description
Use ReportAlign to specify whether the report should be left-justified, right-justified, centered or stretched within the page. These are the possible values of ReportAlign:

Value	Meaning
raLeft	Align report to the left side of the page.
raRight	Align report to the right side of the page.
raCenter	Center report horizontally on the page.
raStretch	Stretch report horizontally on the whole page.
raStretchAll	Stretch all DBGrid's columns horizontally on the one page.

See also: TXQRGrid.ReportPart, TXQRGrid.ReportPartCount

TXQRGrid.ReportBands

Describes all bands in the report.

```
property ReportBands: TXReportBands;
```

Description
The ReportBands represents a container for bands created by XQRGrid component.

See also: TXReportBands

TXQRGrid.ReportDBGrid

Specifies name of data-grid component to use as a base for report settings.

```
property ReportDBGrid: TXDBGrid;
```

Description
Use ReportDBGrid to specify the name of the TXDBGrid component which will be used as a skeleton for the report.

See also: TXDBGrid, TXDBGrid.Report

TXQRGrid.ReportNames

Determines how the report controls are named.

```
type
  TXReportNames = (rnNone, rnClass, rnShort, rnLong);
property ReportNames: TXReportNames;
```

Description
Use ReportNames to specify naming convention for the report controls created by XQRGrid. These are the possible values of ReportNames:

Value	Meaning
rnNone	The report controls have no name.
rnClass	The report controls are named using name of control's class.
rnShort	The report controls are named using base name followed by name of control's class.
rnLong	The report controls are named using name of band and base name followed by name of control's class.

The ReportNames property is availabled in Professional version of X-Files Components only. Standard version of X-Files Components do'nt use names for report controls.

See also: TXQRGrid.ReportTitle

TXQRGrid.ReportPart

Determines the part of report, in multi-part report to print or preview.

```
property ReportPart: Integer;
```

Description
Use ReportPart to specify, which part of report (vertical band) should be print or preview. Multi-part report is created automatically when total width of report exceeds width of one page. Succeeding parts of report are numbered from 1 to ReportPartCount.

See also: TXQRGrid.ReportPartCount, TXQRGrid.CalcReportPartCount, TXQRGrid.ReportAlign

TXQRGrid.ReportPartCount

Specifies the number of parts, in a multi-part report.

```
property ReportPartCount: Integer;
```

Description
Use ReportPartCount to read the number of parts (vertical bands) that should be created for complete report.

ReportPartCount is a read-only property.

See also: TXQRGrid.ReportPart, TXQRGrid.CalcReportPartCount, TXQRGrid.ReportAlign

TXQRGrid.ReportStyle

Represents the report specific options.

```
property ReportStyle: TXReportStyle;
```

Description
Use the ReportStyle property to set report specific options. See TXReportStyle for more infomation.

See also: TXReportStyle

TXQRGrid.ReportTitle

Specifies the title for the report.

```
property ReportTitle: string;
```

Description
Use ReportTitle to specify a title for the report. The ReportTitle is used as a caption for preview window and as a title for the print job. See TQuickRep.ReportTitle for more information.

See also: TXQRGrid.ReportNames, TCustomQuickRep.ReportTitle

TXQRGrid.StripeColor

Specifies the alternate background color to draw horizontal striped rows.

```
property StripeColor: TColor;
```

Description
Set StripeColor to alternate color to draw horizontal striped rows in the report. The selected color should be alternate to main Color value, which is always used as color of second stripe.

The StripeColor property can handle two special colors: clDefault, clNone.
Select clDefault value to use TXDBGrid.StripeColor value.
Select clNone value when the report should not be striped.

See also: TXCustomDBGrid.StripeColor

TXQRGrid.CalcPageWidth

Calculates printer page with to fit to the width of printed grid.

```
procedure CalcPageWidth;
```

Description
Use CalcPageWidth method to calculate PrinterPage.Width suitable to print whole ReportDBGrid on single vertical band. CalcPageWidth method always changes PrinterPage.PaperSize to Custom. This procedure is useful to show or export report as single vertical band, but the result may not be suitable to print report on the printer. See also: roAutoCalcPageWidth option.

See also: TXQRGrid.Options, TXQRGrid.PrinterPage

TXQRGrid.CalcReportPartCount

Recalculates the ReportPartCount property.

```
function CalcReportPartCount: Integer;
```

Description
Use CalcReportPartCount method to recalculate the ReportPartCount property. By default the ReportPartCount property is calculated only when report is created by TXQRGrid component or when ReportPartCount property is first time read. In other case, you should call function CalcReportPartCount to retrieve new value of ReportPartCount.

See also: TXQRGrid.ReportPart, TXQRGrid.ReportPartCount

TXQRGrid.Create

Creates and initializes an instance of TXQRGrid.

```
constructor Create(AOwner: TComponent);
```

Description
Call Create for a TXQRGrid to create the component at runtime. TXQRGrid placed on a form at design time are created automatically. After calling the inherited constructor, Create creates the helper objects used by the component and initializes

- Options to [roFirstPageHeader, roLastPageFooter, roPrintIfEmpty, roShowProgress, roForceTrueTypes, roAllowFixedColor, roModalMode].
- ReportAlign to raStretch.
- ReportNames to rnShort (Professional version only).

• ReportPart to 1.
• ReportTitle to 'X-Files Components Report'.

See also: TXQRGrid.Destroy

TXQRGrid.CreateReport

Creates the TQuickRep object that stores the report definition.

```
type
  TXReportForm = TForm;
function CreateReport(var ReportForm: TXReportForm): TQuickRep; virtual;
```

Description
Call CreateReport to create a report using ReportDBGrid as the skeleton. The TQuickRep object and all his controls created by CreateReport function are owned by ReportForm created also by CreateReport function. If you call directly CreateReport function, you should also release ReportForm using ReportForm.Release.

In many things you not need using CreateReport directly. Instead of CreateReport you may call Print, Preview, SaveReport, ShowReport method, that calls CreateReport internally and destroys its after using.

See also: TXQRGrid.OnReportCreate

TXQRGrid.Destroy

Destroys an instance of TXQRGrid.

```
destructor Destroy; override;
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free verifies that the component is not nil, and only then calls Destroy. Destroy frees all the helper objects used by the TXQRGrid.

See also: TXQRGrid.Create

TXQRGrid.IsPrepareNeeded

Determines whether the report is prepared before printing or preview.

```
function IsPrepareNeeded: Boolean;
```

Description
When PrepareNeeded is rvTrue or when PrapareNeeded is rvAuto and at least one of SysData contains counter, IsPrepareNeeded function return True. Otherwise IsPrepareNeeded return False.

See also: TXQRGrid.PrepareNeeded, TXQRGrid.AfterPrepare

TXQRGrid.Preview

Displays the report on the screen.

```
procedure Preview;
```

Description
Use Preview to generate the report and display it on the screen. See TQuickRep.Preview for more information. The Preview method is especially useful for designers and is accessible from TXQRGrid component's popup menu or by double click on component's icon.

See also: TXQRGrid.Print, TXQRGrid.PrintAll, TCustomQuickRep.Preview

TXQRGrid.Print

Generates the report to the printer.

```
procedure Print;
```

Description
Call Print to generate the report to the printer. See TQuickRep.Print for more information.

See also: TXQRGrid.Preview, TXQRGrid.PrintAll, TCustomQuickRep.Print

TXQRGrid.PrintAll

Generates all parts of the report to the printer.

```
procedure PrintAll;
```

Description
Call PrintAll to generate all parts (vertical bands) of the report to the printer. The PrintAll method modifies ReportPart property from 1 to ReportPartCount and call internally Print method for each part of the report.

See also: TXQRGrid.Preview, TXQRGrid.Print, TCustomQuickRep.Print

TXQRGrid.PrinterSetup

Shows the Printer Setup dialog for the report.

```
procedure PrinterSetup;
```

Description
Use PrinterSetup method to show Printer Setup dialog for the report. When roKeepPrinterSetup and/or roKeepDesignPrinterSetup option is included in Options, the changes made in Printer Setup dialog are saved to PrinterPage and PrinterSettings property. The PrinterSetup method is especially useful for designers and it's accessible from TXQRGrid component's popup menu.

See also: TXQRGrid.PrinterPage, TXQRGrid.PrinterSettings

TXQRGrid.SaveReport

Saves the report definition to the file specified in FileName.

```
procedure SaveReport(const FileName: string);
```

Description
Use the SaveReport method to store report definition to the file(s) specified in FileName parameter. The SaveReport method stores complete Delphi (ReportUnit.pas + ReportUnit.dfm) or C++Builder (ReportUnit.cpp + ReportUnit.h + ReportUnit.dfm) module which can be opened in Delphi or C++Builder IDE and added to the project as standalone unit. When FileName extension is ".pas" SaveReport creates the Delphi module. When FileName extension is ".cpp" SaveReport creates the C++Builder module.

Before you add the saved report to the project, you must change XReportForm.Name, because this class name is internaly used by TXQRGrid. The SaveReport method is especially useful for designers and is accessible from TXQRGrid component's popup menu.

The SaveReport method is available only in registered version of X-Files Components.

See also: TXQRGrid.ShowReport

TXQRGrid.SaveToFile

Saves the content of the report to the file specified in FileName.

```
procedure SaveToFile(const FileName: string; FilterClass: TQRExportFilterClass
= nil);
```

Description
Use the SaveToFile method to save the content of the report to the file specified in FileName parameter with using FilterClass export filter. The SaveToFile method can save content of report in following formats dependent on FilterClass parameter:

Value	Format		QuickReport unit	
nil	QR file	(*.qrp)		
TQRAsciiExportFilter	Text file	(*.txt)	QRExport	- QR 3.x Std.
TQRCommaSeparatedFilter	CSV file	(*.csv)	QRExport	- QR 3.x Std.
TQRHTMLDocumentFilter	HTML file	(*.html)	QRExport	- QR 3.x Std.
TQRGHTMLDocumentFilter	HTML file	(*.html)	QRWebFilt	- QR 3.6 Pro.
TQRXLSFilter	Excel file	(*.xls)	QRExport	- QR 3.6 Pro.
TQRRTFExportFilter	RTF file	(*.rtf)	QRExport	- QR 3.6 Pro.
TQRWMFExportFilter	WMF file	(*.wmf)	QRExport	- QR 3.6 Pro.
TQRPDFDocumentFilter	PDF file	(*.pdf)	QRPDFFilt	- QR 4.0 Pro.
TQRXDocumentFilter	QRX file	(*.qrx)	QRXMLSFilt	- QR 4.0 Pro.

See also: TXQRGrid.SaveReport

TXQRGrid.ShowReport

Shows the report definition as a modal form.

```
procedure ShowReport;
```

Description
Use ShowReport to show the report definition as a modal form. The ShowReport method is especially useful for designers and it's accessible from TXQRGrid component's popup menu.

See also: TXQRGrid.SaveReport

TXQRGrid.AfterBandPrint

Occurs after a band is printed.

```
property AfterBandPrint: TQRBandAfterPrintEvent;
```

Description
The AfterBandPrint event is called after a band is printed. See TQRBand.AfterPrint for more information.

See also: TXQRGrid.AfterPagePrint, TXQRGrid.BeforeBandPrint, TQRCustomBand.AfterPrint

TXQRGrid.AfterPagePrint

Occurs after a page is printed.

```
property AfterPagePrint: TQRNotifyEvent;
```

Description
The AfterPagePrint event is called after a page is printed. See TQuickRep.OnEndPage for more information.

See also: TXQRGrid.AfterBandPrint, TXQRGrid.BeforePagePrint, TCustomQuickRep.OnEndPage

TXQRGrid.AfterPrepare

Occurs when XQRGrid finishes prepared report.

```
property AfterPrepare: TQRNotifyEvent;
```

Description
The AfterPrepare event is called when the grid finishes prepared report. See TQuickRep.Prepare for more information.

See also: TXQRGrid.AfterPrint, TXQRGrid.BeforePrint, TXQRGrid.AfterPreview

TXQRGrid.AfterPreview

Occurs after the user closes the preview form.

```
property AfterPreview: TQRAfterPreviewEvent;
```

Description
The AfterPreview event is called when the user closes the preview form. See TQuickRep.AfterPreview for more information.

See also: TXQRGrid.AfterPrint, TXQRGrid.BeforePrint, TCustomQuickRep.AfterPreview

TXQRGrid.AfterPrint

Occurs when printing of the report is finished.

```
property AfterPrint: TQRAfterPrintEvent;
```

Description
The AfterPrint event is called when printing of the report is finished. See TQuickRep.AfterPrint for more information.

See also: TXQRGrid.AfterPreview, TXQRGrid.BeforePrint, TCustomQuickRep.AfterPrint

TXQRGrid.BeforeBandPrint

Occurs before a band will be printed.

```
property BeforeBandPrint: TQRBandBeforePrintEvent;
```

Description
The BeforeBandPrint event is called before a band will be printed. See TQRBand.BeforePrint for more information.

See also: TXQRGrid.BeforePagePrint, TXQRGrid.AfterBandPrint, TQRCustomBand.BeforePrint

TXQRGrid.BeforePagePrint

Occurs before a page will be printed.

```
property BeforePagePrint: TQRNotifyEvent;
```

Description
The BeforePagePrint event is called before new page will be printed. See TQuickRep.OnStartPage for more information.

See also: TXQRGrid.BeforeBandPrint, TXQRGrid.AfterPagePrint, TCustomQuickRep.OnStartPage

TXQRGrid.BeforePrint

Occurs before the report is generated.

```
property BeforePrint: TQRReportBeforePrintEvent;
```

Description
The BeforePrint event is called before the report is generated. See TQuickRep.BeforePrint for more information.

See also: TXQRGrid.AfterPreview, TXQRGrid.AfterPrint, TCustomQuickRep.BeforePrint

TXQRGrid.OnPrint

Occurs when printable controls will be printed.

```
type
  TQRPrintEvent = procedure(Sender: TObject; var Value: string) of object;
property OnPrint: TQRPrintEvent;
```

Description
The OnPrint event is called when a control like TQRLabel, TQRDBText, TQRExpr or TQRSysData will be printed. The Sender parameter identifies the control to be printed. You can change the text by changing the Value parameter. See TQRLabel.OnPrint for more information.

See also: TQRCustomLabel.OnPrint

TXQRGrid.OnReportCreate

Occurs when the report definition is created by TXQRGrid.

```
property OnReportCreate: TQRNotifyEvent;
```

Description
The OnReportCreate event is called when creating of the report definition is finished by CreateReport method. Write OnReportCreate event handler to extend report definition by adding custom bands at runtime. Additional memory allocated in this event should be released in OnReportDestroy event handler. Using of this futures is recommended only for advanced QuickReport's users.

See also: TXQRGrid.CreateReport, TXQRGrid.OnReportDestroy

TXQRGrid.OnReportDestroy

Occurs when the report is destroyed by TXQRGrid.

```
property OnReportDestroy: TQRNotifyEvent;
```

Description
The OnReportDestroy event is called when the report created by CreateReport is already destroyed. Write OnReportDestroy event handler to free additional memory allocated in OnReportCreate event handler. Using of this futures is recommended only for advanced QuickReport's users.

See also: TXQRGrid.OnReportCreate

TXQRPreviewSettings

Represents initial settings of QuickReport's preview form.

Unit
XQRGrids

Description
TXQRPreviewSettings holds initial settings of QuickReport's preview form. TXQRPreviewSettings possibilities are accesible starting from QuickReport 3.6 Professional. For QuickReport 3.x Standard the properties are of no importance. Some properties can be accesible starting from QuickReport 4.0 Professional.

See also: TXQRGrid.PreviewSettings

TXQRPreviewSettings.FormHeight

Specifies initial height of preview form.

```
property FormHeight: Integer;
```

Description
The FormHeight property represents initial height of preview form. When FormHeight is 0, the initail height of preview form is dependent on screen height.

Notice. The FormHeight property is accesible starting from QuickReport 3.6 Professional. For QuickReport 3.x Standard the FormHeight property are of no importance.

See also: TXQRPreviewSettings.FormWidth

TXQRPreviewSettings.FormWidth

Specifies initial width of preview form.

```
property FormWidth: Integer;
```

Description
The FormWidth property represents initial width of preview form. When FormWidth is 0, the initail width of preview form is dependent on screen width.

Notice. The FormWidth property is accesible starting from QuickReport 3.6 Professional. For QuickReport 3.x Standard the FormWidth property are of no importance.

See also: TXQRPreviewSettings.FormHeight

TXQRPreviewSettings.ShowSearch

Determines the search system is available in preview form.

```
property ShowSearch: Boolean;
```

Description
The ShowSearch property determines the search system is available in preview form.

Notice. The ShowSearch property is accesible starting from QuickReport 4.0 Professional. For QuickReport 3.x Standard and Professional the ShowSearch property are of no importance.

See also: TXQRPreviewSettings.ShowThumbs

TXQRPreviewSettings.ShowThumbs

Determines the thumbs are available in preview form.

```
property ShowThumbs: Boolean;
```

Description
The ShowThumbs property determines the thumbs are available in preview form.

Notice. The ShowThumbs property is accesible starting from QuickReport 4.0 Professional. For QuickReport 3.x Standard and Professional the ShowThumbs property are of no importance.

See also: TXQRPreviewSettings.ShowSearch

TXQRPreviewSettings.WindowState

Represents how the preview form appears on the screen.

```
property WindowState: TWindowState;
```

Description
The WindowState property represents initial state of preview form window. Set WindowState to minimize or maximize the preview form window.

Notice. The WindowState property is accesible starting from QuickReport 3.6 Professional. For QuickReport 3.x Standard the WindowState property are of no importance.

See also: TXQRPreviewSettings.ZoomState

TXQRPreviewSettings.ZoomState

Represents how the report appears in preview form.

```
type
  TXQRZoomState = (zsFullPage, zsPageWidth, zsOriginalSize);
property ZoomState: TXQRZoomState;
```

Description
The ZoomState property represents initial state of report in preview form. These are the possible values of ZoomState:

Value	Meaning
zsFullPage	Whole page of report is visible in preview form.
zsPageWidth	The report is scaled to the width of preview form.
zsOriginalSize	The report has original size (100 %).

Notice. The ZoomState property is accesible starting from QuickReport 4.0 Professional. For QuickReport 3.x Standard and Professional the ZoomState property are of no importance.

See also: TXQRPreviewSettings.WindowState

TXQRPreviewSettings.Create

Creates and initializes an instance of TXQRPreviewSettings.

```
constructor Create;
```

Description
The Create constructor creates an instance of TXQRPreviewSettings. The user never need to call this method directly.

See also: TXQRPreviewSettings

TXReportBands

TXReportBands represents a container for TXReportBand objects.

Unit
XQRGrids

Description
TXReportBands holds descriptions of all QuickReport bands created by TXQRGrid.

See also: TXReportBand

TXReportBands.GridBands

Specifies common settings for report controls in all grid's bands.

```
property GridBands: TXReportBandStyle;
```

Description
The GridBands represents a container for common settings for report controls in all grid's bands.

See also: TXReportBands.GridColumns, TXReportBands.GridHeaders, TXReportBands.GridTitles, TXReportBands.GridTotals

TXReportBands.GridColumns

Specifies custom settings for report controls which represent grid's columns.

```
property GridColumns: TXReportGridBand;
```

Description
The GridColumns represents a container for custom settings for report controls which represents XDBGrid's columns.

See also: TXReportBands.GridBands, TXReportBands.GridHeaders, TXReportBands.GridTitles, TXReportBands.GridTotals

TXReportBands.GridHeaders

Specifies custom settings for report controls which represent grid's headers.

```
property GridHeaders: TXReportGridBand;
```

Description
The GridHeaders represents a container for custom settings for report controls which represents XDBGrid's headers.

See also: TXReportBands.GridBands, TXReportBands.GridColumns, TXReportBands.GridTitles, TXReportBands.GridTotals

TXReportBands.GridTitles

Specifies custom settings for report controls which represent grid's titles.

```
property GridTitles: TXReportGridBand;
```

Description
The GridTitles represents a container for custom settings for report controls which represents XDBGrid's titles.

See also: TXReportBands.GridBands, TXReportBands.GridColumns, TXReportBands.GridHeaders, TXReportBands.GridTotals

TXReportBands.GridTotals

Specifies custom settings for report controls which represent grid's totals.

```
property GridTotals: TXReportGridBand;
```

Description
The GridTotals represents a container for custom settings for report controls which represents XDBGrid's totals. The totals of XDBGrid's columns can be specified in Report property.

See also: TXReportBands.GridBands, TXReportBands.GridColumns, TXReportBands.GridHeaders, TXReportBands.GridTitles

TXReportBands.GroupExpression

Determines when the group bands are printed.

```
property GroupExpression: string;
```

Description
The arithmetic GroupExpression is evaluated on every record and determines when the group bands should be printed. See TQRGroup.Expression to read more. See also roGroupForceNewPage option.

See also: TXReportBands.GroupHeader, TXReportBands.GroupFooter, TXReportBands.GroupTotals

TXReportBands.GroupFooter

Specifies settings for data printed in group footer.

```
property GroupFooter: TXReportGroupBand;
```

Description
The GroupFooter represents a container for properties which specify text of group footer. This band is printed only when GroupExpression is defined.

See also: TXReportBands.GroupHeader, TXReportBands.GroupTotals, TXReportBands.GroupExpression

TXReportBands.GroupHeader

Specifies settings for data printed in group header.

```
property GroupHeader: TXReportGroupBand;
```

Description
The GroupHeader represents a container for properties which specify text of group header. This band is printed only when GroupExpression is defined.

See also: TXReportBands.GroupFooter, TXReportBands.GroupTotals, TXReportBands.GroupExpression

TXReportBands.GroupTotals

Specifies custom settings for report controls which represent group totals.

```
property GroupTotals: TXReportGridBand;
```

Description
The GroupTotals represents a container for custom settings for report controls which represents group totals. The totals of XDBGrid's columns can be specified in Report property. This band is printed only when GroupExpression is defined.

See also: TXReportBands.GroupHeader, TXReportBands.GroupFooter, TXReportBands.GroupExpression

TXReportBands.PageFooter

Specifies settings for data printed in page footer.

```
property PageFooter: TXReportPageBand;
```

Description
The PageFooter represents a container for properties which specify text and layout of page footer.

See also: TXReportBands.PageHeader, TXReportBands.PageLegend, TXReportBands.PageTitle

TXReportBands.PageHeader

Specifies settings for data printed in page header.

```
property PageHeader: TXReportPageBand;
```

Description
The PageHeader represents a container for properties which specify text and layout of page header.

See also: TXReportBands.PageFooter, TXReportBands.PageLegend, TXReportBands.PageTitle

TXReportBands.PageLegend

Specifies settings for data printed in legend of the report.

```
property PageLegend: TXReportPageBand;
```

Description
The PageLegend represents a container for properties which specify text and layout of report's legend. The legend of the report is printed below totals of the columns and under page footer. If Options includes roAllPagesLegend, the PageLegend band is printed on all pages, otherwise the band is printed only on the last page.

See also: TXReportBands.PageFooter, TXReportBands.PageHeader, TXReportBands.PageTitle

TXReportBands.PageTitle

Specifies settings for data printed in title of the report.

```
property PageTitle: TXReportPageBand;
```

Description
The PageTitle represents a container for properties which specify text and layout of report's title. If Options includes roAllPagesTitle, the PageTitle band is printed on all pages, otherwise the band is printed only on the first page.

See also: TXReportBands.PageFooter, TXReportBands.PageHeader, TXReportBands.PageLegend

TXReportBands.Create

Creates and initializes an instance of TXReportBands.

```
constructor Create;
```

Description
The Create constructor creates an instance of TXReportBands and all helper objects. The user never need to call this method directly.

See also: TXReportBands.Destroy

TXReportBands.Destroy

Destroys an instance of TXReportBands.

```
destructor Destroy; override;
```

Description
Destroy frees the helper objects of the TXReportBands before destroying the instance. The user never need to call this method directly.

See also: TXReportBands.Create

TXReportBand

TXReportBand represents single band in a report created by TXQRGrid.

Unit
XQRGrids

Description
TXReportBand introduces main properties for any band in a report created by TXQRGrid.

Do not create instances of TXReportBand. Use TXReportBand as a base class when declaring custom grid bands. TXReportBand is a base class for TXReportGridBand and TXReportPageBand.

See also: TXReportBands, TXReportGridBand, TXReportPageBand

TXReportBand.Color

Specifies the background color for all report controls in the band.

```
property Color: TColor;
```

Description
The Color property determines the background color for all report controls in the band. You can set Color to one of the constants defined in the Graphics unit (such as clBlue), or to an explicit RGB integer value. Set Color to clNone to use default color depend on current ReportDBGrid settings.

See also: TXReportBand.Font

TXReportBand.Font

Specifies the font for all report controls in the band.

```
property Font: TFont;
```

Description
The Font property points to a TFont object that determines typographic attributes of text for all report controls in the band. Set Font.Name to 'Default' to use default font depend on current ReportDBGrid settings. The rest of Font properties are ignored then.

See also: TXReportBand.Color

TXReportBand.Lines

Specifies number of lines for text printed in the band.

```
property Lines: Integer;
```

Description
Set Lines property to override default number of lines for text printed in the band. By default (Lines = 0), number of printed lines is determined by number of visibled lines in the XDBGrid component.

See also: TXReportBand.Visible, TXCustomDBGrid.LinesCount, TXCustomDBGrid.TitleLinesCount, TXCustomDBGrid.HeaderLinesCount

TXReportBand.Name

Specifies the name of the band.

```
property Name: string;
```

Description
The Name property identifies the band in the report. Value for Name is specified when the band is created.

Name is a read-only property.

See also: TXReportBand.QRBand

TXReportBand.QRBand

Identifies the QuickReport's band when the report is created.

```
property QRBand: TQRCustomBand;
```

Description
The QRBand property is used internaly by TXQRGrid to identify QuickReport's band created upon TXReportBand definition. You should using this property in OnReportCreate event handler only.

See also: TXReportBand.Name

TXReportBand.Visible

Determines when the band is visible in the report.

```
property Visible: TXReportVisible;
```

Description
Set Visible property to determine when the band is visible in the report. Set Visible to rvAuto to set default visibility depend on current ReportDBGrid settings.

See also: TXReportBand.Lines, TXReportVisible

TXReportBand.Create

Creates and initializes an instance of TXReportBand.

```
constructor Create(const AName: string);
```

Description
The Create constructor creates an instance of TXReportBand and all helper objects and initializes

- Color to clNone.
- Font.Color to clBlack.
- Font.Name to 'Default'.
- Name to AName.
- Visible to rvAuto.

See also: TXReportBand.Destroy

TXReportBand.Destroy

Destroys an instance of TXReportBand.

```
destructor Destroy; override;
```

Description
Destroy frees the helper objects of the TXReportBand before destroying the instance.

See also: TXReportBand.Create

TXReportGridBand

TXReportGridBand represents grid's band in a report created by TXQRGrid.

Unit
XQRGrids

Description
TXReportGridBand represents XDBGrid's band in the report. TXReportGridBand publishes many of the properties inherited from TXReportBand, but does not introduce any new behavior.

See also: TXReportBands, TXReportBand

TXReportTextBand

TXReportTextBand represents text band in a report created by TXQRGrid.

Unit
XQRGrids

Description
TXReportTextBand inherits main posibilities from TXReportBand and introduce new Text property. TXReportTextBand is an ancestor for TXReportGroupBand and TXReportPageBand.

See also: TXReportBands, TXReportBand

TXReportTextBand.Text

Specifies the text that appears within the band.

```
property Text: TXReportText;
```

Description
The Text represents a container for captions that appears within the band.

See also: TXReportText

TXReportTextBand.Create

Creates and initializes an instance of TXReportTextBand.

```
constructor Create(const AName: string);
```

Description
The Create constructor creates an instance of TXReportTextBand and all helper objects, calls inherited Create constructor and initializes Color to clWhite.

See also: TXReportTextBand.Destroy

TXReportTextBand.Destroy

Destroys an instance of TXReportTextBand.

```
destructor Destroy; override;
```

Description
Destroy frees the helper objects of the TXReportTextBand before destroying the instance.

See also: TXReportTextBand.Create

TXReportGroupBand

TXReportGroupBand represents header's or footer's group band in a report created by TXQRGrid.

Unit
XQRGrids

Description
TXReportGroupBand inherits all posibilities from TXReportTextBand.

See also: TXReportBands, TXReportTextBand

TXReportPageBand

TXReportPageBand represents header's or footer's band in a report created by TXQRGrid.

Unit
XQRGrids

Description
TXReportPageBand inherits main posibilities from TXReportTextBand and introduce new properties which specify layout of page's headers and footers.

See also: TXReportBands, TXReportTextBand

TXReportPageBand.BandExtend

Specifies a margin on the bottom side of the band.

```
property BandExtend: Integer;
```

Description
BandExtend specifies, in pixels, the margin that separates the bottom side of the band from the controls within it.

See also: TXReportPageBand.BandIndent

TXReportPageBand.BandIndent

Specifies a margin on the top side of the band.

```
property BandIndent: Integer;
```

Description
BandIndent specifies, in pixels, the margin that separates the top side of the band from the controls within it.

See also: TXReportPageBand.BandExtend

TXReportPageBand.Border

Specifies kind of the border drawn around the controls within a band.

```
type
  TXReportBorder = (rbNone, rbSpace, rbFrame, rbTopLine, rbBottomLine);
property Border: TXReportBorder;
```

Description
Set the Border property to determine kind of border drawn around the controls. These are the possible values of Border:

Value	Meaning
rbNone	Nothing is drawn around the controls.
rbSpace	Invisible frame is drawn around the controls.
rbFrame	Rectangular frame is drawn around the controls.
rbTopLine	Horizontal line is drawn over the controls.
rbBottomLine	Horizontal line is draw under the controls.

See also: TXReportText

TXReportPageBand.Create

Creates and initializes an instance of TXReportPageBand.

```
constructor Create(const AName: string; AExtend, AIndent: Integer; ABorder: TXReportBorder);
```

Description
The Create constructor creates an instance of TXReportPageBand, calls inherited Create constructor and initializes

- BandExtend to AExtend.
- BandIndent to AIndent.
- Border to ABorder.

See also: TXReportPageBand.Destroy

TXReportBandStyle

TXReportBandStyle represents common properties for all grid's bands.

Unit
XQRGrids

Description
TXReportBandStyle introduces common properties for all grid's bands in a report created by TXQRGrid. Setting non-default values for any properties in this class override XDBGrid's settings that are used as default for a report creation.

See also: TXReportStyle, TXReportBands.GridBands

TXReportBandStyle.Color

Specifies the background color that overrides default color for controls in all grid's bands.

```
property Color: TColor;
```

Description
The Color property overrides default background color for controls in all grid's bands. You can set Color to one of the constants defined in the Graphics unit (such as clBlue), or to an explicit RGB integer value. Set Color to clNone to return to default color depend on current ReportDBGrid settings.

See also: TXReportBandStyle.FontColor

TXReportBandStyle.FontCharset

Specifies the character set of the font that overrides default character set for controls in all grid's bands.

```
property FontCharset: TFontCharset;
```

Description
The FontCharset property overrides default character set of the font for controls in all grid's bands. Each typeface (specified by the Font.Name property) supports one or more character sets. Check the information supplied by the font vendor to determine what values of FontCharset are valid. Set FontCharset to DEFAULT_CHARSET to return to default charcter set depend on current ReportDBGrid settings.

See also: TXReportBandStyle.FontColor, TXReportBandStyle.FontName, TXReportBandStyle.FontSize

TXReportBandStyle.FontColor

Specifies the color of the font that overrides default color of the text characters for controls in all grid's bands.

```
property FontColor: TColor;
```

Description
The FontColor property overrides default color of fonts for controls in all grid's bands. Set FontColor to clNone to return to default color of the text characters depend on current ReportDBGrid settings.

See also: TXReportBandStyle.Color, TXReportBandStyle.FontCharset, TXReportBandStyle.FontName, TXReportBandStyle.FontSize

TXReportBandStyle.FontName

Specifies the typeface of the font that overrides default typeface of fonts for controls in all grid's bands.

```
property FontName: TFontName;
```

Description
The FontName property overrides default typeface of fonts for controls in all grid's bands. Set FontName to 'Default' to return to default typeface of fonts depend on current ReportDBGrid settings.

See also: TXReportBandStyle.FontCharset, TXReportBandStyle.FontColor, TXReportBandStyle.FontSize

TXReportBandStyle.FontSize

Specifies the height of the font in points that overrides default height of fonts for controls in all grid's bands.

```
property FontSize: Integer;
```

Description
The FontSize property overrides default height of fonts for controls in all grid's bands. Set FontSize to 0 to return to default height of fonts depend on current ReportDBGrid settings.

See also: TXReportBandStyle.FontCharset, TXReportBandStyle.FontColor, TXReportBandStyle.FontName

TXReportBandStyle.Create

Creates and initializes an instance of TXReportBandStyle.

```
constructor Create;
```

Description
The Create constructor creates an instance of TXReportBandStyle and initializes

- Color to clNone.
- FontCharset to DEFAULT_CHARSET.
- FontColor to clNone.
- FontName to 'Default'.
- FontSize to 0.

See also: TXReportBands.GridBands

TXReportText

TXReportText represents left, center and right captions in TXReportTextBand.

Unit
XQRGrids

Description
TXReportText introduces properties for captions that appears within the band.

See also: TXReportPageBand.Text

TXReportText.Caption

Specifies the captions that appear in the band.

```
property Caption[Index: Integer]: string;
```

Description
The Caption property contains text strings that appear at the left side (Index=0), at the right side (Index=1) and in the centre (Index=2) of the band. Instead Caption array property you may use LeftCaption, RightCaption or CenterCaption property. All specified captions are drawn transparently. Each Caption may follow a appropriate SysData.

See also: TXReportText.SysData, TXReportText.CenterCaption, TXReportText.LeftCaption, TXReportText.RightCaption

TXReportText.CenterCaption

Specifies the text that is centered in the band.

```
property CenterCaption: string;
```

Description
The CenterCaption property contains a text string that is centered in the band. See Caption property for more information.

See also: TXReportText.Caption, TXReportText.LeftCaption, TXReportText.RightCaption

TXReportText.CenterSysData

Specifies the system information that is centered in the band.

```
property CenterSysData: TXReportSysData;
```

Description
The CenterSysData property contains the system information that is centered in the band. See SysData property for more information.

See also: TXReportText.SysData, TXReportText.LeftSysData, TXReportText.RightSysData

TXReportText.LeftCaption

Specifies the text that appears at the left side of the band.

```
property LeftCaption: string;
```

Description
The LeftCaption property contains a text string that appears at the left side of the band. See Caption property for more information.

See also: TXReportText.Caption, TXReportText.CenterCaption, TXReportText.RightCaption

TXReportText.LeftSysData

Specifies the system information that appears at the left side of the band.

```
property LeftSysData: TXReportSysData;
```

Description
The LeftSysData property contains the system information that appears at the left side of the band. See SysData property for more information.

See also: TXReportText.SysData, TXReportText.CenterSysData, TXReportText.RightSysData

TXReportText.QRField

Identifies the QuickReport's control when the report is created.

```
property QRField[Index: Integer]: TQRCustomLabel;
```

Description
The QRField property is used internaly by TXQRGrid to identify QuickReport's control created upon Caption and SysData definition. You may use this property only inside XQRGrid's event handler.

See also: TXReportText.Caption, TXReportText.SysData

TXReportText.RightCaption

Specifies the text that appears at the right side of the band.

```
property RightCaption: string;
```

Description
The RightCaption property contains a text string that appears at the right side of the band. See Caption property for more information.

See also: TXReportText.Caption, TXReportText.CenterCaption, TXReportText.LeftCaption

TXReportText.RightSysData

Specifies the system information that appears at the right side of the band.

```
property RightSysData: TXReportSysData;
```

Description
The RightSysData property contains the system information that appears at the right side of the band. See SysData property for more information.

See also: TXReportText.SysData, TXReportText.CenterSysData, TXReportText.LeftSysData

TXReportText.SysData

Specifies the system informations that appear in the band.

```
type
  TXReportSysData = (rdTime, rdDate, rdDateTime, rdPageNumber, rdReportTitle,
    rdDetailCount, rdDetailNo, rdNone, rdPartNo, rdPartCount,
    rdPartNoPerCount, rdExpression, rdGrowExpression, rdRecordNo,
    rdRecordCount, rdRecordNoPerCount, rdPageCount, rdPageNoPerCount,
    rdCustomNo, rdCustomCount, rdCustomNoPerCount);
property SysData[Index: Integer]: TXReportSysData;
```

Description
The SysData property contains system informations that appear at the left side (Index=0), at the right side (Index=1) and in the centre (Index=2) of the band. Instead SysData array property you may use LeftSysData, RightSysData or CenterSysData property. All specified system informations are drawn transparently. Each SysData may be followed by appropriate Caption. These are the possible values of SysData:

Value	Meaning
rdTime	Current time.
rdDate	Current date.
rdDateTime	Current date and time.
rdPageNumber	Number of page that is printed.
rdReportTitle	Value of ReportTitle property.
rdDetailCount	Count of records in the DataSet.
rdDetailNo	Number of record that is printed.
rdNone	Nothing.
rdPartNo	Number of vertical band that is printed.
rdPartCount	Count of vertical bands in the report.

rdPartNoPerCount	Number of vertical band per count of bands.
rdExpression	The QuickReport's expression specified in Caption. Value of expression is ResetAfterPrint.
rdGrowExpression	The QuickReport's expression specified in Caption. Value of expression is NOT ResetAfterPrint.
rdRecordNo	Ordinal position of printed record (RecordNumber).
rdRecordCount	Count of printed records (RecordCount).
rdRecordNoPerCount	Ordinal position of printed record per count of printed records.
rdPageCount	Count of printed pages (PageCount).
rdPageNoPerCount	Number of page that is printed per count of printed pages.
rdCustomNo	Number of record selected by developer (CustomNumber).
rdCustomCount	Count of records selected by developer (CustomCount).
rdCustomNoPerCount	Number of record per count of records selected by developer.

See also: TXReportText.Caption, TXReportText.CenterSysData, TXReportText.LeftSysData, TXReportText.RightSysData

TXReportText.Create

Creates and initializes an instance of TXReportText.

```
constructor Create;
```

Description
The Create constructor creates an instance of TXReportText and initializes

• Caption array to ''.
• SysData array rdNone.

See also: TXReportPageBand.Text

TXReportStyle

TXReportStyle represents common properties for all controls in the report created by TXQRGrid.

Unit
XQRGrids

Description
TXReportStyle introduces common properties for all controls in the report created by TXQRGrid. Each TXReportStyle's property specifies report specific option that determines a report layout.

See also: TXQRGrid.ReportStyle

TXReportStyle.ColLines

Determines whether the lines between columns are visibled.

```
property ColLines: TXReportVisible;
```

Description
Set ColLines to rvTrue to cause the lines between columns are visibled. Set ColLines to rvFalse to hide the lines between columns. Set ColLines to rvAuto to cause the lines between columns are visibled only, when Options in ReportDBGrid includes dgColLines.

See also: TXReportStyle.RowLines, TXReportStyle.LineColor, TXReportStyle.LineWidth

TXReportStyle.LineColor

Determines the color used to draw lines in the report.

```
property LineColor: TColor;
```

Description
Set LineColor to change the color used to draw lines.

See also: TXReportStyle.ColLines, TXReportStyle.RowLines, TXReportStyle.LineWidth

TXReportStyle.LineWidth

Specifies the width of the lines.

```
property LineWidth: Integer;
```

Description
Set LineWidth to change width of lines for the report. Set LineWidth to 0 to hide all lines.

See also: TXReportStyle.ColLines, TXReportStyle.RowLines, TXReportStyle.LineColor

TXReportStyle.RowLines

Determines whether the lines between the rows are visibled.

```
property RowLines: TXReportVisible;
```

Description
Set RowLines to rvTrue to cause the lines between the rows are visibled. Set RowLines to rvFalse to hide the lines between rows. Set RowLines to rvAuto to cause the lines between the rows are visibled only, when Options in ReportDBGrid includes dgRowLines.

See also: TXReportStyle.ColLines, TXReportStyle.LineColor, TXReportStyle.LineWidth

TXReportStyle.TextColExtend

Specifies a margin on the right side of the cell.

```
property TextColExtend: Integer;
```

Description
TextColExtend specifies, in pixels, the margin that separates the right side of the cell from the text within it.

See also: TXReportStyle.TextColIndent, TXReportStyle.TextRowExtend, TXReportStyle.TextRowIndent

TXReportStyle.TextColIndent

Specifies a margin on the left side of the cell.

```
property TextColIndent: Integer;
```

Description
TextColIndent specifies, in pixels, the margin that separates the left side of the cell from the text within it.

See also: TXReportStyle.TextRowExtend, TXReportStyle.TextColExtend, TXReportStyle.TextRowIndent

TXReportStyle.TextRowExtend

Specifies a margin on the bottom side of the cell.

```
property TextRowExtend: Integer;
```

Description
TextRowExtend specifies, in pixels, the margin that separates the bottom side of the cell from the text within it.

See also: TXReportStyle.TextColIndent, TXReportStyle.TextColExtend, TXReportStyle.TextRowIndent

TXReportStyle.TextRowIndent

Specifies a margin on the top side of the cell.

```
property TextRowIndent: Integer;
```

Description
TextRowIndent specifies, in pixels, the margin that separates the top side of the cell from the text within it.

See also: TXReportStyle.TextRowExtend, TXReportStyle.TextColExtend, TXReportStyle.TextColIndent

TXReportStyle.Create

Creates and initializes an instance of TXReportStyle.

```
constructor Create;
```

Description
The Create constructor creates an instance of TXReportStyle and initializes

- LineColor to clBlack.
- LineWidth to 2.
- TextColIndent to 1.
- TextColExtend to 1.
- TextRowIndent to 1.
- TextRowExtend to 0.
- ColLines to rvAuto.
- RowLines to rvAuto.

See also: TXQRGrid.ReportStyle

TXReportVisible type

Determines whether an element is visibled in the report.

Unit
XQRGrids

```
type
  TXReportVisible = (rvFalse, rvTrue, rvAuto);
```

Description
The following are possible values of TXReportVisible:

Value	Meaning
rvFalse	Element is never visibled in the report.
rvTrue	Element is always visibled in the report.
rvAuto	Element is visibled in the report if it's just visibled in ReportDBGrid component.

See also: TXReportBand.Visible, TXReportStyle.ColLines, TXReportStyle.RowLines

TXDBColumnsDialog

TXDBColumnsDialog displays a column-selection dialog.

Unit
XDBLists

Description
TXDBColumnsDialog displays a modal dialog box for selecting columns. The dialog does not appear at runtime until it is activated by a call to the Execute method. When the user clicks OK or Apply, the dialog closes and the selected columns are marked as "Visible" in the linked DBGrid component. See also DialogOptions.

See also: TXDBPrintColumnsDialog, TXColumn.DialogOptions

TXDBColumnsDialog.BorderStyle

Determines whether the dialog has a single line border around the columns list.

```
property BorderStyle: TBorderStyle;
```

Description
Use BorderStyle to affect the sharpness. BorderStyle can have a value of either bsSingle or bsNone. If BorderStyle is bsSingle, the dialog has a single-line border around the columns list. If BorderStyle is bsNone, there will be no border.

See also: TXDBColumnsDialog.BorderStyle

TXDBColumnsDialog.ButtonsTitles

Specifies the titles that appear on the buttons.

```
property ButtonsTitles: string;
```

Description
Use ButtonsTitles to specify the titles that appear on the buttons in dialog box. Each caption is separated by semicolon. The default value for ButtonsTitles is `'OK;Cancel;&Apply;&Help'`.

See also: TXDBColumnsDialog.ColumnsTitle, , TXDBColumnsDialog.SortedTitle, TXDBColumnsDialog.Title

TXDBColumnsDialog.ColumnsTitle

Specifies the title caption for the columns list.

```
property ColumnsTitle: string;
```

Description
Use ColumnsTitle to set the title caption for the columns list.

See also: TXDBColumnsDialog.ButtonsTitles, , TXDBColumnsDialog.SortedTitle, TXDBColumnsDialog.Title

TXDBColumnsDialog.DBGrid

Specifies name of data-grid component to use in columns dialog.

```
property DBGrid: TXDBGrid;
```

Description
Use DBGrid to specify the name of the TXDBGrid component which columns should appear on list in dialog box.

See also: TXDBGrid

TXDBColumnsDialog.DefaultCount

Specifies the default count of list items that are visibled in the dialog box.

```
property DefaultCount: Integer;
```

Description
Use DefaultCount to specify count of list items visibled in the dialog box. DefaultCount can't be less then MinimumCount.

See also: TXDBColumnsDialog.DefaultWidth, TXDBColumnsDialog.MinimumCount

TXDBColumnsDialog.DefaultWidth

Specifies the default width for the dialog box.

```
property DefaultWidth: Integer;
```

Description
Use DefaultWidth to specify width of dialog box. If DefaultWidth is 0, width of dialog box is depend on buttons that are visibled in dialog box. DefaultWidth can't be less then MinimumWidth.

See also: TXDBColumnsDialog.DefaultCount, TXDBColumnsDialog.MinimumWidth

TXDBColumnsDialog.HelpContext

Specifies the context number for online Help

```
property HelpContext: THelpContext;
```

Description
The HelpContext property is an integer value that determines which Help screen appears when the user requests context-sensitive online Help.

See also: TXDBColumnsDialog.Options

TXDBColumnsDialog.MinimumCount

Specifies the minimum count of list items that are visibled in the dialog box.

```
property MinimumCount: Integer;
```

Description
Use MinimumCount to specify then minimum count of list items visibled in the dialog box. MinimumCount must be less then DefaultCount and can't be less then 4 items.

See also: TXDBColumnsDialog.MinimumWidth, TXDBColumnsDialog.DefaultCount

TXDBColumnsDialog.MinimumWidth

Specifies the minimum width for the dialog box when it's resized.

```
property MinimumWidth: Integer;
```

Description
Use MinimumWidth to specify then minimum width of dialog box. If MinimumWidth is 0, the minimum width of dialog box is depend on buttons that are visibled in dialog box. MinimumWidth must be less then DefaultWidth.

See also: TXDBColumnsDialog.MinimumCount, TXDBColumnsDialog.DefaultWidth

TXDBColumnsDialog.Options

Determines the appearance and behavior of the column-selection dialog.

```
type
  TXDBColumnsOption = (coAllowMoving, coAutoUpdate, coCancelButton,
    coApplyButton, coHelpButton, coMarkPrintedCols, coMarkVisibleCols,
    coShowEmptyCols, coShowFixedCols, coShowHeaders, coShowSorted); {* ver.
    6.1 *}
  TXDBColumnsOptions = set of TXDBColumnsOption;
property Options: TXDBColumnsOptions;
```

Description
Use the Options property to customize the appearance and functionality of the dialog. See also DialogOptions property. The possible values of Options are:

Value	Meaning
coAllowMoving	User can move items on columns list.
coAutoUpdate	All changes made on columns list are automatically moved to the DBGrid component.
coCancelButton	Displays a Cancel button in the dialog.
coApplyButton	Displays an Apply button in the dialog.
coHelpButton	Displays a Help button in the dialog.
coMarkPrintedCols	Columns Visibled in the report are initialy checked on columns list.
coMarkVisibleCols	Columns Visibled in the DBGrid are initialy checked on columns list.
coShowEmptyCols	Columns with empty FieldName property also appear on columns list.
coShowFixedCols	Fixed columns appear on columns list.
coShowHeaders	Columns captions are followed by headers captions.
coShowSorted	Columns list can be sorted. Sorted check box is visible. {* ver. 6.1 *}

See also: TXDBColumnsDialog.DBGrid, TXColumn.DialogOptions, TXColumnReport.DialogOptions

TXDBColumnsDialog.SortedTitle

Specifies the title caption for the sorted check box.

```
property SortedTitle: string; {* ver. 6.1 *}
```

Description
Use SortedTitle to set the caption for the sorted check box.

See also: TXDBColumnsDialog.ButtonsTitles, TXDBColumnsDialog.ColumnsTitle, TXDBColumnsDialog.Title

TXDBColumnsDialog.Title

Specifies the title caption for the dialog box.

```
property Title: string;
```

Description
Use Title to set the title caption for the dialog box.

See also: TXDBColumnsDialog.ButtonsTitles, TXDBColumnsDialog.ColumnsTitle, TXDBColumnsDialog.SortedTitle

TXDBColumnsDialog.Create

Creates and initializes a TXDBColumnsDialog instance.

```
constructor Create(AOwner: TComponent);
```

Description
Call Create to instantiate a columns dialog component at runtime. Dialogs added to forms at design time are created automatically. The dialog box does not appear on the screen at runtime until the Execute method is called.

See also: TXDBColumnsDialog.DBGrid

TXDBColumnsDialog.Execute

Displays the column-selection dialog.

```
function Execute: Boolean;
```

Description
Execute opens the column-selection dialog, returning True when the user clicks OK. If the user clicks Cancel, Execute returns False.

After the dialog box opens captions of DBGrid's columns appear on list. User can click on checkbox to select or unselect a column. User can also move a column to other place by using drag & drop or by using arrows with Shift key. When the user clicks OK or Apply button the changes made in column-selection dialog are moved to the DBGrid component. When the user clicks Cancel button no changes are made in DBGrid component.

The current functionality of column-selection dialog depend on Options property.

See also: TXDBColumnsDialog.Options

TXDBColumnsDialog.OnApply

Occurs when the user clicks OK or Apply button.

```
property OnApply: TNotifyEvent;
```

Description
Use an OnApply event handler to apply changes the user made in additional controls added to the dialog box in OnCreate event handler. The OnApply event occurs when changes should be stored in DBGrid component.

The functions maintained by this dialog are stored automatically before OnApply event occurs. When coAutoUpdate option is included in Options the OnApply event handler is called every time when the user made any changes. See Demo1 example to know more.

Notice! As Sender is passed dialog form (TXDBColumnsForm).

See also: TXDBColumnsDialog.OnCreate, TXDBColumnsDialog.OnClose, TXDBColumnsDialog.OnShow,

TXDBColumnsDialog.OnClose

Occurs when the dialog closes.

```
property OnClose: TNotifyEvent;
```

Description
Write an OnClose event handler to perform special processing when the dialog closes.

Notice! As Sender is passed dialog form (TXDBColumnsForm).

See also: TXDBColumnsDialog.OnCreate, TXDBColumnsDialog.OnApply, TXDBColumnsDialog.OnShow,

TXDBColumnsDialog.OnCreate

Occurs when a XDBColumnsDialog component instantiates a dialog box.

```
property OnCreate: TNotifyEvent;
```

Description
Write an OnCreate event handler to take specific actions when a XDBColumnsDialog component instantiates a dialog box. For example, you may add in run-time a custom controls to the CustomPanel in the dialog box. See Demo1 example to know more.

Notice! As Sender is passed dialog form (TXDBColumnsForm).

See also: TXDBColumnsDialog.OnApply, TXDBColumnsDialog.OnClose, TXDBColumnsDialog.OnShow,

TXDBColumnsDialog.OnShow

Occurs when the dialog opens.

```
property OnShow: TNotifyEvent;
```

Description
Write an OnShow event handler to perform special processing when the dialog box is displayed. For example, you can made any changes on columns list. See XFilesDemo1 example to know more.

Notice! As Sender is passed dialog form (TXDBColumnsForm).

See also: TXDBColumnsDialog.OnCreate, TXDBColumnsDialog.OnApply, TXDBColumnsDialog.OnClose,

TXDBPrintColumnsDialog

TXDBPrintColumnsDialog displays a print-column-selection dialog.

Unit
XDBLists

Description
TXDBPrintColumnsDialog displays a modal dialog box for selecting columns. The dialog does not appear at runtime until it is activated by a call to the Execute method. When the user clicks OK, the dialog closes and the selected columns are marked as "Printed" in the linked DBGrid component. See also DialogOptions.

See also: TXDBColumnsDialog, TXColumnReport.DialogOptions

TXBlobField

TXBlobField represents an extended TBlobField.

Unit
XDBFields

Description
TXBlobField extends functionality of TBlobField to achieve assignment compatibility between BLOB field and TPicture class for another graphic formats.

TBlobField supports only TBitmap graphic assignment. TXBlobField can assign from TPicture (and AssignTo TPicture) additionally TIcon, TMetafile, TJPEGImage and any other TGraphic class descendant recognized by GraphicClass.

The new functionality allows directly using TImage, TDBImage, TQRImage, TQRDBImage with any graphic format stored in BLOB fields (*.bmp, *.ico, *.wmf, *.emf, *.jpg and other supported by GraphicClass).

TXBlobField is the direct ancestor of TXGraphicField.

Notice. To can using TXBlobField as default BLOB field you must have Professional version of the package (with source) recompiled with `{$DEFINE XDBFIELDS}` switch in XDBFields.pas unit. All new BLOB fields added in Fields Editor will be created then as TXBlobField, but you must to replace TBlobField with TXBlobField in existing *.dfm & *.pas files.

See also: TBlobField, TXGraphicField

TXBlobField.Assign

Copies a value to the BLOB field.

```
procedure Assign(Source: TPersistent); override;
```

Description
Use Assign to copy data to the BLOB field. TXBlobField.Assign extends TBlobField.Assign.

TBlobField supports only TBitmap graphic assignment. TXBlobField can assign from TPicture (and AssignTo TPicture) additionally TIcon, TMetafile, TJPEGImage and any other TGraphic class descendant recognized by GraphicClass.

The new functionality allows directly using TImage, TDBImage, TQRImage, TQRDBImage with any graphic format stored in BLOB fields (*.bmp, *.ico, *.wmf, *.emf, *.jpg and other supported by GraphicClass).

See also: TBlobField.Assign

TXBlobField.GraphicClass

Recognizes graphic stored in BLOB field.

```
function GraphicClass: TGraphicClass;
```

Description
GraphicClass function return the class reference for TGraphic appropriate to the current contents of the BLOB field. GraphicClass return result of FieldGraphicClass global function.

See also: FieldGraphicClass

TXGraphicField

TXGraphicField represents an extended TGraphicField.

Unit
XDBFields

Description
TXGraphicField extends functionality of TGraphicField to achieve assignment compatibility between graphic BLOB field and TPicture class for another graphic formats.

TGraphicField supports only TBitmap graphic assignment. TXGraphicField can assign from TPicture (and AssignTo TPicture) additionally TIcon, TMetafile, TJPEGImage and any other TGraphic class descendant recognized by GraphicClass.

The new functionality allows directly using TImage, TDBImage, TQRImage, TQRDBImage with any graphic format stored in graphic BLOB fields (*.bmp, *.ico, *.wmf, *.emf, *.jpg and other supported by GraphicClass). Starting from RAD Studio 2010 the graphic format *.png and *.gif are also supported without using of GraphicEx library. `{* ver. 5.4 *}`

TXGraphicField differs from its immediate ancestor TXBlobField only in having a DataType of ftGraphic.

Notice. To can using TXGraphicField as default graphic BLOB field you must have Professional version of the package (with source) recompiled with `{$DEFINE XDBFIELDS}` switch in XDBFields.pas unit. All new graphic BLOB fields added in Fields Editor will be created then as TXGraphicField, but you must to replace TGraphicField with TXGraphicField in existing *.dfm & *.pas files.

See also: TGraphicField, TXBlobField

TXGraphicField.Create

Creates an instance of TXGraphicField.

```
constructor Create(AOwner: TComponent);
```

Description
Create sets the DataType property to ftGraphic. The AOwner parameter specifies the component, typically a dataset, that becomes the new field's Owner. The Owner is responsible for freeing the component.

See also: TGraphicField.Create

FieldGraphicClass function

Recognizes graphic stored in BLOB field.

Unit
XDBFields

```
function FieldGraphicClass(Field: TField): TGraphicClass;
```

Description
FieldGraphicClass function return the class reference for TGraphic appropriate to the current contents of the BLOB field. When the type of graphic (bitmap, icon, metafile or jpeg) is known, FieldGraphicClass return its specific type object (TBitmap, TIcon, TMetafile or TJPEGImage). Otherwise, FieldGraphicClass return Nil.

TXDBGrid component can directly display graphics (stored in BLOB fields) of any format supported by this function (when DataType of BLOB field is ftGraphic). FieldGraphicClass function supports by default *.bmp, *.ico, *.wmf, *.emf & *.jpg graphics. Starting from RAD Studio 2010 the graphic format *.png and *.gif are also supported without using of GraphicEx library. `{* ver. 5.4 *}`

If you have Professional version of the package, you can recompile the package with `{$DEFINE GraphicEx}` switch in XDBFields unit. It allows you to store and recognize in BLOB field other graphic formats supported by **GraphicEx** library (*.png, *.pcx, *.scr, *.gif, *.tif, *.eps, etc.). This library was written by **Mike Lischke** and is distributed under **Mozilla Public Licence**. You can download **GraphicEx** library from http://www.soft-gems.net/ .

You can also use the functionality of FieldGraphicClass for TImage, TDBImage, TQRImage, TQRDBImagecomponents, but first you must to replace TBlobField & TGraphicField with TXBlobField & TXGraphicField classes.

See also: TXBlobField.GraphicClass

TXFGradient

TXFGradient specifies the gradient drawing style for a control.

Unit
XFSGraph

Description
TXFGradient class is a fundamental class of gradient drawing style. It holds all properties and methods needed to realize gradient drawing for X-Files controls.

See also: TXFGradientButton, TXFGradientBackground, TXFGradientProgress

TXFGradient.Active

Specifies whether or not the gradient drawing should be in use.

```
property Active: Boolean;
```

Description
Use Active to determine or set whether the gradient drawing should be in use. When Active is True (default) gradient drawing may be in use. When Active is False gradient drawing is disabled. See also IsActive function to read when gradient drawing is really in use.

See also: TXFGradient.IsActive

TXFGradient.Cells

Specifies the grid's data cells for gradient drawing.

```
type
  TXFGradientCell = (gcSelectedCell, gcSelectedRow, gcMultiSelected,
    gcStripedRows);
  TXFGradientCells = set of TXFGradientCell;
property Cells: TXFGradientCells; {* ver. 5.0 *}
```

Description
Use the Cells property to customize the appearance of the grid's data cells with using gradient color. The possible values of Cells are:

Value	Meaning
gcSelectedCell	Draw gradient color for selected (focused) cell in the grid. See also: SelectCellColor
gcSelectedRow	Draw gradient color for selected (current) row. See also: SelectRowColor
gcMultiSelected	Draw gradient color for multi selected rows. See also: SelectionColor
gcStripedRows	Draw gradient color for striped rows. See also: StripeColor

See also: TXFGradient.Options

TXFGradient.Direction

Specifies the direction of gradient drawing.

```
type
  TXFGradientDirection = (gdHorizontal, gdVertical);   // For
    C++Builder/Delphi 5, 6, 7
  TXFGradientDirection = GraphUtil.TGradientDirection; // For Delphi 8 or
    higher
property Direction: TXFGradientDirection;
```

Description
Use Direction to determine direction of gradient drawing. When Direction is gdHorizontal the gradient is filling from top to botton of an area. When Direction is gdVertical the gradient is filling from left to right of an area. You can revert the direction of gradient filling (from bottom to top or from right to left) by using SwapColors property.

See also: TXFGradient.SwapColors, TXFGradient.StartColor, TXFGradient.EndColor

TXFGradient.EndColor

Specifies the final color for gradient area.

```
property EndColor: TColor;
```

Description
Set EndColor to determine final color for gradient area when gradient drawing IsActive. The default value of EndColor property is clDefault. It means that original color of the area will be used as gradient EndColor. Change EndColor property only when you want to use different colors for gradient and non-gradient drawing. The initial color of gradient area is determined by StartColor property.

See also: TXFGradient.HotColor, TXFGradient.StartColor, TXFGradient.StepColor

TXFGradient.HotColor

Specifies the color for gradient hot buttons.

```
property HotColor: TColor;
```

Description
Set HotColor to determine hot color for gradient buttons when gradient hot button drawing IsHotActive. The default value of HotColor property is clDefault. It means that DefaultHotColor will be used as gradient HotColor. The initial hot color for gradient hot buttons is determined by StartColor property.

See also: TXFGradient.EndColor, TXFGradient.StartColor, TXFGradient.StepColor, TXFGradient.DefaultHotColor, TXFGradient.IsHotActive

TXFGradient.Options

Determines the appearance of gradient elements.

```
type
  TXFGradientOption = (goGradient, goHotButton, goHotTrack, goHotColor);
  TXFGradientOptions = set of TXFGradientOption;
property Options: TXFGradientOptions;
```

Description
Use the Options property to customize the appearance of gradient elements. The possible values of Options are:

Value	Meaning
goGradient	Draw gradient element.
goHotButton	Draw gradient hot button.
goHotTrack	Draw hot track around element.
goHotColor	Use HotColor for hot button and/or hot track.

See also: TXFGradient.Active, TXFGradient.HotColor, TXFGradient.IsHotActive

TXFGradient.ShadowFactor

Specifies the shadow factor for gradient drawing.

```
property ShadowFactor: Integer; {* ver. 5.1 *}
```

Description
Set ShadowFactor to determine luminance for gradient drawing. The default value of ShadowFactor is -25. The value of ShadowFactor determines the final color of gradient for EndColor, HotColor and StepColor property. Set ShadowFactor to 0, to use directly base color as final color of gradient.

See also: TXFGradient.ShadowColor, TXFGradient.HotColor, TXFGradient.EndColor, TXFGradient.StepColor

TXFGradient.StartColor

Specifies the initial color for gradient area.

```
property StartColor: TColor;
```

Description
Set StartColor to determine initial color for gradient area when gradient drawing IsActive. The default value of StartColor property is clWindow. The final color of gradient area is determined by EndColor, HotColor or StepColor property.

See also: TXFGradient.HotColor, TXFGradient.EndColor, TXFGradient.StepColor

TXFGradient.StepColor

Specifies the color for gradient progress bars.

```
property StepColor: TColor;
```

Description
Set StepColor to determine final color for gradient progress bar when gradient drawing IsActive. The default value of StepColor property is clDefault. It means that original progress bar color will be used as gradient StepColor. Change StepColor property only when you want to use different colors for gradient and non-gradient progress bar drawing. The initial color of gradient progress bar is determined by StartColor property.

See also: TXFGradient.HotColor, TXFGradient.StartColor, TXFGradient.EndColor, TXFGradient.FillStep

TXFGradient.SwapColors

Specifies the direction of filling gradient colors.

```
property SwapColors: Boolean;
```

Description
Use SwapColors to change the direction of filling gradient color. By default gradient colors are filling top/bootom or left/right. Set SwapColors property to fill gradient bottop/top or right/left. See also Direction property.

See also: TXFGradient.Direction, TXFGradient.StartColor, TXFGradient.EndColor

TXFGradient.Assign

Copies a value to the gradient object.

```
procedure Assign(Source: TPersistent); override;
```

Description
Use Assign to copy data from one to another gradient object. The Assign method does not cause OnChange event. See also Change method.

See also: TXFGradient.Change

TXFGradient.Change

Changes a value of the gradient object.

```
function Change(Gradient: TXFGradient): Boolean; virtual;
```

Description
Use Change to copy data from one to another gradient object. The Change method causes OnChange event. See also Assign method.

See also: TXFGradient.Assign, TXFGradient.OnChange

TXFGradient.Create

Creates and initializes a TXFGradient instance.

```
constructor Create(Control: TControl);
```

Description
The Create constructor creates and initializes an instance of TXFGradient class. If you create TXFGradient instance as a property for any control you must specify this Control in parameter to correctly detect Themes mode in design-time since Delphi 2007. See also ThemesEnabled function.

See also: TXFGradient

TXFGradient.DefaultHotColor

Specifies the default color for gradient hot buttons.

```
function DefaultHotColor: TColor; virtual;
```

Description
DefaultHotColor determines default HotColor for gradient buttons when gradient hot button drawing IsHotActive.

See also: TXFGradient.HotColor, TXFGradient.IsHotActive

TXFGradient.FillBack

Fills background of the specified rectangle on the canvas using gradient colors.

```
procedure FillBack(Canvas: TCanvas; Rect: TRect; Color: TColor; BevelWidth,
  SpaceWidth: Integer; HotButton: Boolean = False; DownButton: Boolean =
  False; SoftButton: Boolean = False); virtual;
```

Description
Use FillBack to fill background of a rectangular region on the Canvas using gradient colors. The Rect region decreased by BevelWidth is filled from top to boottom (Direction=gdVertical) or from left to right (Direction=gdHorizontal) with using StartColor and EndColor. When EndColor is clDefault the Color parameter will be used instead EndColor. The Rect region decreased by BevelWidth + SpaceWidth is treats as transparent area. It means, that only the background of this area is filled with using gradient colors. The foreground pixels are not changed. The Color parameter determines the background color.

When HotBotton parameter is True the "hot" Options will be respected. When DownButton is True the shadow color will be used for hot button.

See also: TXFGradient.FillCell, TXFGradient.FillRect, TXFGradient.FillStep, TXFGradient.EndColor, TXFGradient.StartColor

TXFGradient.FillCanvas

Fills the specified rectangle on the canvas using gradient colors.

```
class procedure FillCanvas(Canvas: TCanvas; StartColor, EndColor: TColor;
  const Rect: TRect; Direction: TXFGradientDirection); {* ver. 5.1 *}
```

Description
Use FillCanvas to fill a rectangular region on the Canvas using gradient colors. The Rect region is filled from top to boottom (Direction=gdVertical) or from left to right (Direction=gdHorizontal) with using StartColor and EndColor parameters.

See also: TXFGradient.FillRect

TXFGradient.FillCell

Fills background of the specified rectangle on the canvas using gradient colors.

```
procedure FillCell(Canvas: TCanvas; Rect: TRect; Color: TColor; Flat: Boolean
  = False{* ver. 5.1 *}); virtual; {* ver. 5.0 *}
```

Description
Use FillCell to fill background of a rectangular region on the Canvas using gradient colors. The Rect region is always filled from top to bottom with using Color parameter. When Flat parameter is True, the region is filled with using regular color.

See also: TXFGradient.FillBack, TXFGradient.FillRect, TXFGradient.FillStep

TXFGradient.FillRect

Fills the specified rectangle on the canvas using gradient colors.

```
procedure FillRect(Canvas: TCanvas; Rect: TRect; Color: TColor; HotButton:
  Boolean = False; DownButton: Boolean = False); virtual;
```

Description
Use FillRect to fill a rectangular region on the Canvas using gradient colors. The Rect region is filled from top to boottom (Direction=gdVertical) or from left to right (Direction=gdHorizontal) with using StartColor and EndColor. When EndColor is clDefault the Color parameter will be used instead EndColor.

When HotBotton parameter is True the "hot" Options will be respected. When DownButton is True the shadow color will be used for hot button.

See also: TXFGradient.FillBack, TXFGradient.FillCell, TXFGradient.FillStep, TXFGradient.Options, TXFGradient.EndColor, TXFGradient.StartColor

TXFGradient.FillStep

Fills parts of the specified rectangle on the canvas using gradient colors.

```
procedure FillStep(Canvas: TCanvas; Rect: TRect; Color: TColor; BevelWidth,
  SpaceWidth: Integer; HotButton: Boolean = False; DownButton: Boolean =
  False; SoftButton: Boolean = False); virtual;
```

Description
Use FillStep to fill parts of a rectangular region on the Canvas using gradient colors. The Rect region decreased by BevelWidth is filled from top to boottom (Direction=gdVertical) or from left to right (Direction=gdHorizontal) with using StartColor and StepColor. When StepColor is clDefault the Color parameter will be used instead StepColor. The Rect region decreased by BevelWidth + SpaceWidth is treats as transparent area. It means, that only the parts of this area are filled with using gradient colors. The other pixels are not changed. The Color parameters determines the color of these areas.

When HotBotton parameter is True the "hot" Options will be respected. When DownButton is True the shadow color will be used for hot button.

See also: TXFGradient.FillBack, TXFGradient.FillCell, TXFGradient.FillRect, TXFGradient.StepColor, TXFGradient.StartColor

TXFGradient.FillStyleBack

Fills background of the specified rectangle on the canvas using gradient colors of Custom Style.

```
procedure FillStyleBack(Canvas: TCanvas; Rect: TRect; Color: TColor;
  BevelWidth, SpaceWidth: Integer; HotButton: Boolean = False; DownButton:
  Boolean = False; SoftButton: Boolean = False); virtual; {* ver. 5.1 *} //
  For C++Builder/Delphi XE2 or higher
```

Description
Use FillStyleBack to fill background of a rectangular region on the Canvas using gradient colors of Custom Style. The Rect region decreased by BevelWidth is filled from top to boottom (Direction=gdVertical) or from left to right (Direction=gdHorizontal) with using colors corresponding to selected Custom Style. The Rect region decreased by BevelWidth + SpaceWidth is treats as transparent area. It means, that only the background of this area is filled with using gradient colors. The foreground pixels are not changed. The Color parameter determines the background color.

The HotBotton and DownButton parameters determine "hot" and "pressed" state of gradient colors in StyleServices.

See also: TXFGradient.FillBack, TXFGradient.FillStyleRect

TXFGradient.FillStyleRect

Fills the specified rectangle on the canvas using gradient colors of Custom Style.

```
procedure FillStyleRect(Canvas: TCanvas; Rect: TRect; Color: TColor); virtual;
  {* ver. 5.1 *} // For C++Builder/Delphi XE2 or higher
```

Description
Use FillStyleRect to fill a rectangular region on the Canvas using gradient colors of Custom Style. The Rect region is filled from top to boottom (Direction=gdVertical) or from left to right (Direction=gdHorizontal) with using Color parameter corresponding to the selected Custom Style.

See also: TXFGradient.FillStyleBack, TXFGradient.FillRect

TXFGradient.IsActive

Specifies whether or not the gradient drawing is in use.

```
function IsActive(AllowForThemes: Boolean = False): Boolean;
```

Description
Check IsActive to determine when the gradient drawing is in use. IsActive return True when gradient library IsAvailable, GradientEnabled global variable is True and Active property is True. If AllowForThemes parameter is False the result of this function can be True only when ThemesEnabled is False. When IsActive return False the gradient drawing is currently disabled for the control.

See also: TXFGradient.Active, TXFGradient.IsHotActive, TXFGradient.IsAvailable, GradientEnabled, ThemesEnabled

TXFGradient.IsAvailable

Specifies whether or not the gradient library is available.

```
function IsAvailable: Boolean;
```

Description
Check IsAvailable to determine the gradient library is available. TXFGradient class uses system GradientFill procedure from `msimg32.dll` library. This library may be not available on Windows 95 and Windows NT. In that case IsAvailable function return False and gradient drawing is disabled but the application can properly work.

See also: TXFGradient.IsHotActive, TXFGradient.IsActive

TXFGradient.IsHotActive

Specifies whether or not the gradient hot button is in use.

```
function IsHotActive(AllowForThemes: Boolean = False){* ver. 5.1 *}: Boolean;
```

Description
Check IsHotActive to determine is gradient hot button in use. This function return True when IsActive is True and one (or more) of the following option is included in Options: goHotButton, goHotTrack, goHotColor. When AllowForThemes parameter is True, it means that gradient drawing is used also when ThemesEnabled is True.

See also: TXFGradient.Options, TXFGradient.IsActive

TXFGradient.ShadowColor

Returns the shadow color with a specific factor.

```
class function ShadowColor(Color: TColor; Factor: Integer): TColor; {* ver.
5.1 *}
```

Description
ShadowColor function returns the shadow color with a specific factor. Color parameter represents the base color. Factor parameter represents the luminance. The negative value of Factor determines a darker color. The positive value of Factor determines a brighter color.

See also: TXFGradient.ShadowFactor

TXFGradient.SwitchOptions

Include or exclude Options depending on State.

```
procedure SwitchOptions(Part: TXFGradientOptions; State: Boolean); {* ver. 6.0 *}
```

Description
Use SwitchOptions to include or exclude the Part of Options property. When State is True the Part options are included, otherwise excluded.

See also: TXCustomDBGrid.SwitchOptions, TXFGradient.Options

TXFGradient.OnChange

Occurs when the gradient properties was changed.

```
property OnChange: TNotifyEvent;
```

Description
Write an OnChange event handler to take specific actions when the gradient properties was changed. By default this event points to X-Files controls method to Invalidate control when the gradient properties changed.

See also: TXFGradient.Change

TXFGradientBackground

TXFGradientBackground specifies the gradient drawing style for background of panels and groupboxes.

Unit
XFSGraph

Description
TXFGradientBackground class is derived from TXFGradient fundamental class of gradient drawing style. It holds all properties and methods needed to realize gradient drawing for X-Files panels and groupboxes.

See also: TXFGradientButton, TXFGradient, TXFGradientProgress

TXFGradientButton

TXFGradientButton specifies the gradient drawing style for buttons.

Unit
XFSGraph

Description
TXFGradientButton class is derived from TXFGradient fundamental class of gradient drawing style. It holds all properties and methods needed to realize gradient drawing for X-Files buttons.

See also: TXFGradientBackground, TXFGradient, TXFGradientProgress

TXFGradientProgress

TXFGradientProgress specifies the gradient drawing style for progress bars.

Unit
XFSGraph

Description
TXFGradientProgres class is derived from TXFGradient fundamental class of gradient drawing style. It holds all properties and methods needed to realize gradient drawing for X-Files progress bars.

See also: TXFGradientBackground, TXFGradient, TXFGradientButton

GradientEnabled variable

Determines gradient drawing style using for all X-Files controls.

Unit
XFSGraph

```
var
  GradientEnabled: Boolean = True;
```

Description
GradientEnabled global variable determines using gradient drawing style for all X-Files controls in whole application. By default gradient drawing style is enabled. You can set GradientEnabled to False in main module of application to suppress gradient drawing in whole application. When GradientEnabled is False the IsActive function always return False for each TXFGradient object.

See also: ThemesEnabled, TXFGradient.IsActive

IsVista function

Determines Windows Vista (or newer) platform.

Unit
XFSGraph

```
function IsVista: Boolean;
```

Description
IsVista function determines Windows Vista platform. The result is True when you run an application on Windows Vista (or newer) platform otherwise the function returns False.

See also: ThemesEnabled

StylesEnabled function

Determines using Custom Styles in an application.

Unit
XFSGraph

```
function StylesEnabled: Boolean; inline; {* ver. 5.1 *} // For
C++Builder/Delphi XE2 or higher
```

Description
StylesEnabled function determines using Custom Styles in an application. This function is available starting from RAD Studio XE2. Result of StylesEnabled is depend on current result of TStyleManager.IsCustomStyleActive function and `{$DEFINE STYLES}` conditional switch using to build the package. When `{$UNDEF STYLES}` switch was defined, StylesEnabled function always return False.

You can change `{$DEFINE STYLES}` switch in Conditionals.pas include file and rebuild package when you have the professional version of the package. By default this switch is defined starting from RAD Studio XE2.

See also: ThemesEnabled

ThemesEnabled function

Determines using Windows Themes for specified control.

Unit
XFSGraph

```
function ThemesEnabled(Control: TControl = nil): Boolean;
```

Description
ThemesEnabled function determines using Windows Themes for specified control. When Control parameter is nil the result of ThemesEnabled is depend on current result of ThemeServices.ThemesEnabled function and `{$DEFINE THEMES}` conditional switch using to build the package. When `{$UNDEF THEMES}` switch was defined, ThemesEnabled function always return False.

Since Delphi 2007 the Control parameter is required to obtain correct result in design-time for specified Control. The result depends on "Enable runtime themes" check box state in Project Options -> Application settings. In run-time or when Control parameter is nil the function always returns result of ThemeServices.ThemesEnabled function.

You can change `{$UNDEF THEMES}` switch in Conditionals.pas include file and rebuild package when you have the professional version of the package. By default this switch is undefined for Delphi/C++Builder 5 & 6 and defined for Delphi 7 or higher.

See also: GradientEnabled, TXFGradient.IsActive

TButtonDrawingStyle type

Specifies using drawing style for a button.

Unit
XFSGraph

```
type
  TButtonDrawingStyle = (bdsClassic, bdsThemed, bdsGradient); {* ver. 5.1 *}
```

Description
The following are possible values of TButtonDrawingStyle:

Value	Meaning
bdsClassic	The button control uses the standard, unthemed style.
bdsThemed	The button control uses the current operating system theme.
bdsGradient	The button control uses gradients for styling.

See also: TGridDrawingStyle

TFineTheme type

Specifies using additional themes under Windows XP.

Unit
XFSGraph

```
type
  TFineTheme = (ftNoChange, ftBorder, ftUnderline);
```

Description
The following are possible values of TFineTheme:

Value	Meaning
ftNoChange	Do not change default theme.
ftBorder	Use ToolButton's theme with yellow border for hot state.
ftUnderline	Use ToolButton's theme with yellow underline for hot state.

See also: ThemesEnabled

ClearBackground function

Clears background of the control.

Unit
XFSGraph

```
function ClearBackground(Control: TWinControl; DC: HDC): HGDIOBJ;
```

Description
ClearBackground performs clear background when the Control is paint. You not need to call this function directly.

See also: SetupParentBackground

SetupParentBackground function

Specifies csParentBackground control style.

Unit
XFSGraph

```
procedure SetupParentBackground(Control: TControl; Value: Boolean = True);
```

Description
SetupParentBackground includes (Value=True) or excludes (Value=Fale) csParentBackground attribute to ControlStyle characteristics of the Control. You not need to call this procedure directly.

See also: ClearBackground

ControlNeedsInvalidate function

Determines does the control needs Invalidate for gradient background.

Unit
XFSGraph

```
function ControlNeedsInvalidate(Control: TControl): Boolean;;
```

Description
ControlNeedsInvalidate determines does the Control needs Invalidate when the control's parent is using gradient drawing style. For ThemesEnable=True, the result of this function is True when csParentBackground attribute is included in ControlStyle. For ThemesEnabled=False, the result of this function is True when the Control has Transparent published property or either the Control has Gradient and ParentBackground published property and the value of ParentBackground is True. You not need to call this function directly.

See also: NeedsInvalidate

NeedsInvalidate variable

Points to currently using ControlNeedsInvalidate function.

Unit
XFSGraph

```
type
  TControlNeedsInvalidate = function(Control: TControl): Boolean;
var
  NeedsInvalidate: TControlNeedsInvalidate = ControlNeedsInvalidate;
```

Description
NeedsInvalidate variable points to currently using ControlNeedsInvalidate function. X-Files controls call this function to check does the Control needs Invalidate when the control's parent is using gradient drawing style.

See also: ControlNeedsInvalidate

TXFButton

TXFButton is a gradient push button control.

Unit
XFSCtrls

Description
Use TXFButton to put a gradient push button on a form. TXFButton is derrived from the standard TButton control. TXFButton introduces several properties to control border and gradient style.

To use a gradient button that displays a bitmap instead of a label, use TXFBitBtn. To use a gradient button that can remain in a depressed position, use TXFSpeedButton.

See also: TButton, TXFBitBtn, TXFSpeedButton, TXFUpDown

TXFButton.ApplyTheme

Determines whether the Windows Theme is used for the button.

```
property ApplyTheme: Boolean; {* ver. 5.1 *}
```

Description
Set ApplyTheme to true, to draw themed button with using Windows Themes. This property is respected only when XP Manifest is added to the application.

See also: TXFButton.DrawingStyle

TXFButton.DrawingStyle

Specifies the drawing style of the button.

```
property DrawingStyle: TButtonDrawingStyle; {* ver. 5.1 *}
```

Description
Use DrawingStyle to select drawing style for the button. The DrawingStyle is a master property for ApplyThemeand Gradient.Active property.

See also: TXFButton.ApplyTheme, TXFButton.Gradient

TXFButton.Fine

Determines whether the button has a semi-flat or semi-3D border that provides a raised or lowered look.

```
property Fine: Boolean;
```

Description
Set Fine to true to change the raised border when the button is unselected and the lowered border when the button is clicked. The Fine property introduces two new borders (semi-flat or semi-3D) depend on value of Flat property. This property is respected only when the Gradient IsActive.

See also: TXFButton.Flat, TXFButton.Gradient

TXFButton.Flat

Determines whether the button has a 3D border that provides a raised or lowered look.

```
property Flat: Boolean;
```

Description
Set Flat to true to remove the raised border when the button is unselected and the lowered border when the button is clicked. This property is respected only when the Gradient IsActive. See also Fine property.

See also: TXFButton.Fine, TXFButton.Gradient

TXFButton.Gradient

Specifies the gradient drawing style for the button.

```
property Gradient: TXFGradient;
```

Description
The Gradient property holds TXFGradientButton object that determines gradient drawing style for the button. The gradient drawing style IsActive for button only when ThemesEnabled is False.

See also: TXFButton.Fine, TXFButton.Flat

TXFButton.WordWrap

Specifies whether the button text wraps to fit the width of the control.

```
property WordWrap: Boolean;
```

Description
Set WordWrap to true to allow the label to display multiple line of text. When WordWrap is true, text that is too wide for the control wraps at the right margin.

Set WordWrap to false to limit the label to a single line. When WordWrap is false, text that is too wide for the label appears truncated.

This property is now effective in all the versions of Delphi/C++Builder supported by X-Files Components.

See also: TXFBitBtn.WordWrap

TXFButton.Create

Creates and initializes a TXFButton instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a gradient button. Buttons added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the button instance. It becomes the value of the Owner property.

See also: TXFButton.Destroy

TXFButton.Destroy

Destroys the gradient push button object.

```
destructor Destroy; override;
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free checks that the gradient push button is not nil, and only then calls Destroy.

See also: TXFButton.Create

TXFButton.OnMouseEnter

Occurs when the mouse pointer moves over the button.

```
property OnMouseEnter: TNotifyEvent;
```

Description
Write an OnMouseEnter event handler to take specific action when the user moves the mouse over the button. For example, you can use this event to change the font color when the mouse is over the button, and then use the OnMouseLeave event to change it back when the mouse moves off the button.

See also: TXFButton.OnMouseLeave

TXFButton.OnMouseLeave

Occurs when the mouse pointer moves off from over the button.

```
property OnMouseLeave: TNotifyEvent;
```

Description
Write an OnMouseLeave event handler to take specific action when the user moves the mouse off the button. For example, you can use this event to undo changes that were made in an OnMouseEnter event handler.

See also: TXFButton.OnMouseEnter

TXFBitBtn

TXFBitBtn is a gradient push button control that can include a bitmap on its face.

Unit
XFSCtrls

Description
Use TXFBitBtn to put a gradient bitmap button on a form. TXFBitBtn is derrived from the standard TBitBtn control. TXFBitBtn introduces several properties to control border and gradient style.

To use a gradient button that not displays a bitmap, use TXFButton. To use a gradient button that can remain in a depressed position, use TXFSpeedButton.

See also: TBitBtn, TXFButton, TXFSpeedButton, TXFUpDown

TXFBitBtn.ApplyTheme

Determines whether the Windows Theme is used for the button.

```
property ApplyTheme: Boolean; {* ver. 5.1 *}
```

Description
Set ApplyTheme to true, to draw themed button with using Windows Themes. This property is respected only when XP Manifest is added to the application.

See also: TXFBitBtn.DrawingStyle

TXFBitBtn.DrawingStyle

Specifies the drawing style of the button.

```
property DrawingStyle: TButtonDrawingStyle; {* ver. 5.1 *}
```

Description
Use DrawingStyle to select drawing style for the button. The DrawingStyle is a master property for ApplyThemeand Gradient.Active property.

See also: TXFBitBtn.ApplyTheme, TXFBitBtn.Gradient

TXFBitBtn.Fine

Determines whether the button has a semi-flat or semi-3D border that provides a raised or lowered look.

```
property Fine: Boolean;
```

Description
Set Fine to true to change the raised border when the button is unselected and the lowered border when the button is clicked. The Fine property introduces two new borders (semi-flat or semi-3D) depend on value of Flat property. This property is respected only when the Gradient IsActive.

See also: TXFBitBtn.Flat, TXFBitBtn.Gradient

TXFBitBtn.Flat

Determines whether the button has a 3D border that provides a raised or lowered look.

```
property Flat: Boolean;
```

Description
Set Flat to true to remove the raised border when the button is unselected and the lowered border when the button is clicked. This property is respected only when the Gradient IsActive. See also Fine property.

See also: TXFBitBtn.Fine, TXFBitBtn.Gradient

TXFBitBtn.Gradient

Specifies the gradient drawing style for the button.

```
property Gradient: TXFGradient;
```

Description
The Gradient property holds TXFGradientButton object that determines gradient drawing style for the button. The gradient drawing style IsActive for button only when ThemesEnabled is False.

See also: TXFBitBtn.Fine, TXFBitBtn.Flat

TXFBitBtn.WordWrap

Specifies whether the button text wraps to fit the width of the control.

```
property WordWrap: Boolean;
```

Description
Set WordWrap to true to allow the label to display multiple line of text. When WordWrap is true, text that is too wide for the control wraps at the right margin.

Set WordWrap to false to limit the label to a single line. When WordWrap is false, text that is too wide for the label appears truncated.

This property works now effective in all the versions of Delphi/C++Builder supported by X-Files Components.

See also: TXFButton.WordWrap

TXFBitBtn.Create

Creates and initializes a TXFBitBtn instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a gradient bitmap button. Buttons added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the button instance. It becomes the value of the Owner property.

See also: TXFBitBtn.Destroy

TXFBitBtn.Destroy

Destroys the gradient bitmap button object.

```
destructor Destroy; override;
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free checks that the gradient bitmap button is not nil, and only then calls Destroy.

See also: TXFBitBtn.Create

TXFBitBtn.OnMouseEnter

Occurs when the mouse pointer moves over the button.

```
property OnMouseEnter: TNotifyEvent;
```

Description
Write an OnMouseEnter event handler to take specific action when the user moves the mouse over the button. For example, you can use this event to change the font color when the mouse is over the button, and then use the OnMouseLeave event to change it back when the mouse moves off the button.

See also: TXFBitBtn.OnMouseLeave

TXFBitBtn.OnMouseLeave

Occurs when the mouse pointer moves off from over the button.

```
property OnMouseLeave: TNotifyEvent;
```

Description
Write an OnMouseLeave event handler to take specific action when the user moves the mouse off the button. For example, you can use this event to undo changes that were made in an OnMouseEnter event handler.

See also: TXFBitBtn.OnMouseEnter

TXFSpeedButton

TXFSpeedButton is a gradient button that is used to execute commands or set modes.

Unit
XFSCtrls

Description
Use TXFSpeedButton to add a gradient button to a group of buttons in a form. TXFSpeedButton is derrived from the standard TSpeedButton control. TXFSpeedButton introduces several properties to control border and gradient style.

To use a gradient push button that displays a bitmap, use TXFBitBtn. To use a gradient button that not displays a bitmap, use TXFButton.

See also: TSpeedButton, TXFButton, TXFBitBtn, TXFUpDown

TXFSpeedButton.ApplyTheme

Determines whether the Windows Theme is used for the button.

```
property ApplyTheme: Boolean; {* ver. 5.1 *}
```

Description
Set ApplyTheme to true, to draw themed button with using Windows Themes. This property is respected only when XP Manifest is added to the application.

See also: TXFSpeedButton.DrawingStyle, TXFSpeedButton.FineTheme

TXFSpeedButton.DrawingStyle

Specifies the drawing style of the button.

```
property DrawingStyle: TButtonDrawingStyle; {* ver. 5.1 *}
```

Description
Use DrawingStyle to select drawing style for the button. The DrawingStyle is a master property for ApplyThemeand Gradient.Active property.

See also: TXFSpeedButton.ApplyTheme, TXFSpeedButton.Gradient

TXFSpeedButton.Fine

Determines whether the button has a semi-flat or semi-3D border that provides a raised or lowered look.

```
property Fine: Boolean;
```

Description
Set Fine to true to change the raised border when the button is unselected and the lowered border when the button is clicked. The Fine property introduces two new borders (semi-flat or semi-3D) depend on value of Flat property. This property is respected only when the Gradient IsActive.

See also: TSpeedButton.Flat, TXFSpeedButton.FineTheme, TXFSpeedButton.Gradient

TXFSpeedButton.FineTheme

Determines whether the button has a special look under Windows XP.

```
property FineTheme: TFineTheme;
```

Description
Set FineTheme to true to change the default theme under Windows XP. The FineTheme property introduces two new themes when the Fine property is true. When the Fine property is False or either FineTheme is ftNoChange the default themes leave unchanged.

See also: TXFSpeedButton.Fine

TXFSpeedButton.Gradient

Specifies the gradient drawing style for the button.

```
property Gradient: TXFGradient;
```

Description
The Gradient property holds TXFGradientButton object that determines gradient drawing style for the button. The gradient drawing style IsActive for button only when ThemesEnabled is False and Transparent property is not effective.

See also: TXFSpeedButton.Fine, TSpeedButton.Flat, TSpeedButton.Transparent

TXFSpeedButton.Create

Creates and initializes a TXFSpeedButton instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a gradient button. Buttons added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the button instance. It becomes the value of the Owner property.

See also: TXFSpeedButton.Destroy

TXFSpeedButton.Destroy

Destroys the gradient button object.

```
destructor Destroy; override;
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free checks that the gradient button is not nil, and only then calls Destroy.

See also: TXFSpeedButton.Create

TXFSpeedButton.OnMouseEnter

Occurs when the mouse pointer moves over the button.

```
property OnMouseEnter: TNotifyEvent;
```

Description
Write an OnMouseEnter event handler to take specific action when the user moves the mouse over the button. For example, you can use this event to change the font color when the mouse is over the button, and then use the OnMouseLeave event to change it back when the mouse moves off the button.

See also: TXFSpeedButton.OnMouseLeave

TXFSpeedButton.OnMouseLeave

Occurs when the mouse pointer moves off from over the button.

```
property OnMouseLeave: TNotifyEvent;
```

Description
Write an OnMouseLeave event handler to take specific action when the user moves the mouse off the button. For example, you can use this event to undo changes that were made in an OnMouseEnter event handler.

See also: TXFSpeedButton.OnMouseEnter

TXFLabel

TXFLabel is a nonwindowed control that displays text on a gradient background.

Unit
XFSCtrls

Description
Use TXFLabel to display text on a gradient background. TXFLabel is derrived from the standard TLabel control, but Transparent property is default True.

See also: TLabel, TXFStaticText

TXFLabel.Transparent

Specifies whether the background of the caption is transparent.

```
property Transparent: Boolean;
```

Description
Use Transparent to specify whether the background of the caption is transparent. This property is **default** True.

See also: TXFStaticText.Transparent, TXFCheckBox.Transparent, TXFRadioButton.Transparent

TXFLabel.Create

Creates and initializes a TXFLabel instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a transparent label. Labels added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the label instance. It becomes the value of the Owner property.

See also: TLabel.Create

TXFCustomStaticText

TXFCustomStaticText is the base class for TXFStaticText.

Unit
XFSCtrls

Description
TXFCustomStaticText is the base class for TXFStaticText, a windowed control that displays text on a gradient background.

See also: TCustomStaticText, TXFStaticText

TXFCustomStaticText.Transparent

Specifies whether the background of the caption is transparent.

```
property Transparent: Boolean;
```

Description
Use Transparent to specify whether the background of the caption is transparent. This property is **default** True and works effective on gradient background.

See also: TXFLabel.Transparent, TXFCheckBox.Transparent, TXFRadioButton.Transparent

TXFCustomStaticText.Create

Creates and initializes a TXFCustomStaticText instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a transparent static text. Static text added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the static text instance. It becomes the value of the Owner property.

See also: TCustomStaticText.Create

TXFStaticText

TXFStaticText is a windowed control that displays text on a gradient background.

Unit
XFSCtrls

Description
Use TXFStaticText to display text on a gradient background. TXFStaticText is similar to the standard TStaticText control, but Transparent property works effective on a gradient background.

See also: TStaticText, TXFLabel

TXFCustomCheckBox

TXFCustomCheckBox is the ancestor of all transparent check-box components.

Unit
XFSCtrls

Description
TXFCustomCheckBox is the base class for all transparent check-box components. TXFCustomCheckBox is derrived from the standard TCustomCheckBox control, but works effecitve on gradient background.

See also: TCustomCheckBox, TXFCheckBox

TXFCustomCheckBox.Transparent

Specifies whether the background of the check box is transparent.

```
property Transparent: Boolean;
```

Description
Use Transparent to specify whether the background of the caption is transparent. This property is **default** True and works effective on gradient background.

See also: TXFLabel.Transparent, TXFStaticText.Transparent, TXFRadioButton.Transparent

TXFCustomCheckBox.Create

Creates and initializes a TXFCustomCheckBox instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a transparent check box. Check box added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the check box instance. It becomes the value of the Owner property.

See also: TCustomCheckBox.Create

TXFCheckBox

TXFCheckBox represents a transparent check box.

Unit
XFSCtrls

Description
Use TXFCheckBox to display check box on a gradient background. TXFCheckBox is similar to the standard TCheckBox control, but works effecive on gradient background.

See also: TCheckBox, TXFRadioButton

TXFRadioButton

TXFRadioButton represents a transparent radio button.

Unit
XFSCtrls

Description
Use TXFRadioButton to display radio button on a gradient background. TXFRadioButton is similar to the standard TRadioButton control, but works effecive on gradient background.

See also: TRadioButton, TXFCheckBox

TXFRadioButton.Transparent

Specifies whether the background of the radio button is transparent.

```
property Transparent: Boolean;
```

Description
Use Transparent to specify whether the background of the caption is transparent. This property is **default** True and works effective on gradient background.

See also: TXFLabel.Transparent, TXFStaticText.Transparent, TXFCheckBox.Transparent

TXFRadioButton.Create

Creates and initializes a TXFRadioButton instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a transparent radio button. Radio button added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the radio button instance. It becomes the value of the Owner property.

See also: TRadioButton.Create

TXFGroupCheckBox

TXFGroupCheckBox represents a check box for a caption of group box.

Unit
XFSCtrls

Description
TXFGroupCheckBox represents a check box for a caption of group box. This class is used by TXFCustomGroupBox.

See also: TXFCustomGroupBox

TXFGroupCheckBox.Action

Specifies the action associated with the check box.

```
type
  TXFCheckBoxAction = (caNone, caDisable, caEnable);
property Action: TXFCheckBoxAction; {* ver. 5.11 *}
```

Description
Action is the action associated with the check box. When check box is Visible and Checked state is changed the associated action is performed. The possible values of Action are:

Value	Meaning
caNone	No special action is performed.
caDisable	When check box is unchecked, all child controls in group box are disabled. When check box is checked, the disabled child controls in group box are re-enabled.
caEnable	When check box is checked, all child controls in group box are enabled. When check box is unchecked, the enabled child controls in group box are re-disabled.

See also: TXFGroupCheckBox.Checked, TXFGroupCheckBox.Visible

TXFGroupCheckBox.Checked

Specifies whether the check box control is checked.

```
property Checked: Boolean; {* ver. 5.11 *}
```

Description
Use Checked to determine whether a check box control is checked.

See also: TXFGroupCheckBox.Action, TXFGroupCheckBox.Visible

TXFGroupCheckBox.Enabled

Controls whether the check box responds to mouse and keyboard.

```
property Enabled: Boolean; {* ver. 5.11 *}
```

Description
Use Enabled to change the availability of the check box to the user. To disable a check box, set Enabled to false. If Enabled is false, the check box ignores mouse and keyboard events. To re-enable the check box, set Enabled to true.

See also: TXFGroupCheckBox.Visible

TXFGroupCheckBox.Visible

Determines whether the check box appears on group box.

```
property Visible: Boolean; {* ver. 5.11 *}
```

Description
Use the Visible property to control the visibility of the check box. If Visible is true, the control appears. If Visible is false, the control is not visible.

See also: TXFGroupCheckBox.Action, TXFGroupCheckBox.Checked, , TXFGroupCheckBox.Enabled

TXFGroupCheckBox.Create

Creates and initializes a TXFGroupCheckBox and associates it with a GroupBox.

```
property Create(AGroupBox: TXFCustomGroupBox): Boolean; {* ver. 5.11 *}
```

Description
Applications do not need to call create to instantiate the checkbox used by groupbox to implement their CheckBox property. GroupBox calls Create from their constructor to create this object. Component writers that implement additional properties for descendants of TXFGroupCheckBox can call Create in derived class.

See also: TXFGroupCheckBox.Create

TXFGroupCheckBox.Destroy

Destroys the group check box object.

```
property Destroy: Boolean; {* ver. 5.11 *}
```

Description
Do not call Destroy directly in an application. The group check box is automatically destroyed by group box.

See also: TXFGroupCheckBox.Destroy

TXFCustomGroupBox

TXFCustomGroupBox is the base class for all gradient group box components.

Unit
XFSCtrls

Description
TXFCustomGroupBox is the class from which all gradient group box components - including TXFGroupBox, TXFRadioGroup, TXFCheckGroup and TXDBRadioGroup- descend.
TXFCustomGroupBox is derrived from the standard TCustomGroupBox control. TXFCustomGroupBox introduces several properties to control gradient style and parent's background.

See also: TCustomGroupBox, TXFGroupBox

TXFCustomGroupBox.CheckBox

Specifies whether the group box has a check box.

```
property CheckBox: TXFGroupCheckBox; {* ver. 5.11 *}
```

Description
The CheckBox property holds TXFGroupCheckBox object that determines check box control for the group box. When check box is visible it appears near a caption of group box. The check box control may be used to perform any action on all child controls for the group box.

See also: TXFCustomGroupBox.OnCheckBoxChanged

TXFCustomGroupBox.Gradient

Specifies the gradient drawing style for the group box.

```
property Gradient: TXFGradient;
```

Description
The Gradient property holds TXFGradientBackground object that determines gradient drawing style for the group box. The gradient drawing style IsActive for group box only when ParentBackground is False.

See also: TXFCustomPanel.Gradient

TXFCustomGroupBox.ParentBackground

Determines whether the group-box uses parent's background.

```
property ParentBackground: Boolean;
```

Description
If ParentBackground is true, the group-box uses the parent's background to draw its own background. ParentBackground has effect for both ThemesEnabled is True or False.

See also: TXFCustomPanel.ParentBackground

TXFCustomGroupBox.PartialBackground

Determines whether the group-box uses partial parent's background.

```
property PartialBackground: Boolean; {* ver. 5.1 *}
```

Description
If PartialBackground is true, the group-box uses the partial parent's background to draw its own background. The PartialBackground property has effect when parent control has not csParentBackground option in ControlStyle (Parent.ParentBackground = False). This means that the property may not work for all parent controls.

Notice. If you want to make sure that the property PartialBackground to be effective, the group-box control should be placed on TForm or on any class derived from TXFCustomGroupBox or TXFCustomPanel with ParentBackground = False.

See also: TXFCustomGroupBox.ParentBackground

TXFCustomGroupBox.Create

Creates and initializes a TXFCustomGroupBox instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a gradient group box. Group box added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the group box instance. It becomes the value of the Owner property.

See also: TXFCustomGroupBox.Destroy

TXFCustomGroupBox.Destroy

Destroys the gradient group box object.

```
destructor Destroy; override;
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free checks that the gradient group box is not nil, and only then calls Destroy.

See also: TXFCustomGroupBox.Create

TXFCustomGroupBox.OnCheckBoxChanged

Occurs immediately after checked state of check box changes.

```
property OnCheckBoxChanged: TNotifyEvent; {* ver. 5.11 *}
```

Description
Write an OnCheckBoxChanged event handler to respond to changes checked state of check box. This event is generated only when check box is visible in group box. If CheckBox.Action is specified the action is performed before the event occurs.

See also: TXFCustomGroupBox.CheckBox

TXFGroupBox

TXFGroupBox represents a gradient group box.

Unit
XFSCtrls

Description
The TXFGroupBox component represents a gradient group box. TXFGroupBox is similar to the standard TGroupBox control, but introduces several properties to control gradient style and parent's background.

See also: TGroupBox, TXFPanel

TXFCustomRadioGroup

TXFCustomRadioGroup is the base class for gradient radio-group components.

Unit
XFSCtrls

Description
TXFCustomRadioGroup is the class from which gradient radio-group components - including TXFRadioGroup and TXDBRadioGroup - descend. TXFCustomRadioGroup is derrived from the TXFCustomGroupBox control. TXFCustomGroupBox introduces several properties to control gradient style and parent's background.

See also: TXFCustomCheckGroup, TXFCustomGroupBox

TXFRadioGroup

TXFRadioGroup represents a gradient group of radio buttons that function together.

Unit
XFSCtrls

Description
The TXFRadioGroup component represents a gradient group of radio buttons. TXFRadioGroup is similar to the standard TRadioGroup control, but introduces several properties to control gradient style and parent's background.

See also: TXFCheckGroup, TRadioGroup

TXFCustomCheckGroup

TXFCustomCheckGroup is the base class for check-group components.

Unit
XFSCtrls

Description
Check groups are collections of check buttons in a single group box. TXFCustomCheckGroup is the class from which gradient check-group components - including TXFCheckGroup - descend. TXFCustomCheckGroup is similar to TXFCustomRadioGroup, but introduces Checked property to maintain state of check buttons. TXFCustomCheckGroup is derrived from the TXFCustomGroupBox control. TXFCustomGroupBox introduces several properties to control gradient style and parent's background.

See also: TXFCustomRadioGroup, TXFCustomGroupBox

TXFCustomCheckGroup.Buttons

Access individual TXFCheckBox objects.

```
property Buttons[Index: Integer]: TXFCheckBox;
```

Description
Buttons provides direct access to the individual TXFCheckBox objects created by the control. Buttons is a read-only property. This means you can access individual TXFCheckBox objects, but you cannot add or delete TXFCheckBox objects. To modify the button list, change the Items property.

Index is the zero-origin button index. If Index is equal to or greater than the number of buttons, an error is raised.

See also: TCustomRadioGroup.Buttons

TXFCustomCheckGroup.Checked

Indicates which check buttons are checked.

```
property Checked: Integer;
```

Description
For each item of the Items list, one bit in the Checked property is set if a check mark appears in the item's check box. Checked corresponds to the set of TCheckBox.Checked state. The first check button corresponds to first bit in Checked. It's useful to edit a value of any set type.

The value of Checked changes at runtime as the user selects check buttons. If you want one or more of the buttons to appear checked when the application starts, assign a set of that buttons to Checked at design time; otherwise, leave Checked set to the default value of 0, which means that no button is checked.

Hint. If you want all of the buttons to appear checked, assign value of -1 to Checked. If you want to assign a value of any set type, you can use typecasting e.g. Checked := Byte(Anchors); Anchors := TAnchors(Byte(Checked));

Checked can save state for first 32 check buttons. If you really need more than 32 check buttons you can set and examine in run-time a Checked property in Buttons array.

See also: TXFCustomCheckGroup.Checked

TXFCustomCheckGroup.ItemIndex

Indicates which check button in the group was lately checked/unchecked.

```
property ItemIndex: Boolean;
```

Description
ItemIndex holds the index of the lately selected check button in the Items list. (The first button is 0.) The value of ItemIndex changes at runtime as the user selects check buttons. You can not assign a value to ItemIndex directly, but you can examine ItemIndex in OnClick event handler. Initially, ItemIndex has default value of -1, which means that no button was checked.

See also: TXFCustomCheckGroup.ItemIndex

TXFCheckGroup

TXFCheckGroup represents a gradient group of check buttons that function together.

Unit
XFSCtrls

Description
The TXFCheckGroup component represents a gradient group of check buttons. TXFCheckGroup is similar to the TRadioGroup control, but introduces Checked property to maintain state of check buttons and several properties to control gradient style and parent's background.

See also: TXFRadioGroup, TRadioGroup

TXFCustomPanel

TXFCustomPanel is the base class for all gradient panel components.

Unit
XFSCtrls

Description
TXFCustomPanel is the class from which all gradient panel components - including TXFPanel- descend. TXFCustomPanel is derrived from the standard TCustomPanel control. TXFCustomPanel introduces several properties to control gradient style and parent's background.

See also: TCustomPanel, TXFCustomGroupBox

TXFCustomPanel.Gradient

Specifies the gradient drawing style for the panel.

```
property Gradient: TXFGradient;
```

Description
The Gradient property holds TXFGradientBackground object that determines gradient drawing style for the panel. The gradient drawing style IsActive for panel only when ParentBackground is False.

See also: TXFCustomGroupBox.Gradient

TXFCustomPanel.ParentBackground

Determines whether the panel uses parent's background.

```
property ParentBackground: Boolean;
```

Description
If ParentBackground is true, the panel uses the parent's background to draw its own background. ParentBackground has effect for both ThemesEnabled is True or False.

See also: TXFCustomGroupBox.ParentBackground

TXFCustomPanel.ShowCaption

Specifies whether to display the caption of the panel control..

```
property ShowCaption: Boolean; {* ver. 5.1 *}
```

Description
Use ShowCaption to specify whether to display the caption of the panel control.

See also:

TXFCustomPanel.Create

Creates and initializes a TXFCustomPanel instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a gradient panel. Panel added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the panel instance. It becomes the value of the Owner property.

See also: TXFCustomPanel.Destroy

TXFCustomPanel.Destroy

Destroys the gradient panel object.

```
destructor Destroy; override;
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free checks that the gradient panel is not nil, and only then calls Destroy.

See also: TXFCustomPanel.Create

TXFPanel

TXFPanel represents a gradient panel.

Unit
XFSCtrls

Description
The TXFPanel component represents a gradient panel. TXFPanel is similar to the standard TPanel control, but introduces several properties to control gradient style and parent's background.

See also: TPanel, TXFGroupBox

TXFUpDown

TXFUpDown is a gradient up-down control.

Unit
XFSCtrls

Description
The TXFUpDown component represents a gradient up-down control. TXFUpDown is similar to the standard TUpDown control, but introduces several properties to control border and gradient style.

See also: TUpDown

TXFUpDown.Fine

Determines whether the up-down control has a semi-flat or semi-3D border that provides a raised or lowered look.

```
property Fine: Boolean;
```

Description
Set Fine to true to change the raised border when the buttons are unselected and the lowered border when the button is clicked. The Fine property introduces two new borders (semi-flat or semi-3D) depend on value of Flat property. This property is respected only when the Gradient IsActive.

See also: TXFUpDown.Flat, TXFUpDown.Gradient

TXFUpDown.Flat

Determines whether the up-down control has a 3D border that provides a raised or lowered look.

```
property Flat: Boolean;
```

Description
Set Flat to true to remove the raised border when the buttons are unselected and the lowered border when the button is clicked. This property is respected only when the Gradient IsActive. See also Fine property.

See also: TXFUpDown.Fine, TXFUpDown.Gradient

TXFUpDown.Gradient

Specifies the gradient drawing style for the up-down control.

```
property Gradient: TXFGradient;
```

Description
The Gradient property holds TXFGradientButton object that determines gradient drawing style for the buttons. The gradient drawing style IsActive for up-down control only when ThemesEnabled is False.

See also: TXFUpDown.Fine, TXFUpDown.Flat

TXFUpDown.Create

Creates and initializes a TXFUpDown instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a gradient up-down control. Up-Down added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the up-down instance. It becomes the value of the Owner property.

See also: TXFUpDown.Destroy

TXFUpDown.Destroy

Destroys the gradient up-down object.

```
destructor Destroy; override;
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free checks that the gradient up-down control is not nil, and only then calls Destroy.

See also: TXFUpDown.Create

TXFUpDown.OnMouseEnter

Occurs when the mouse pointer moves over the control.

```
property OnMouseEnter: TNotifyEvent;
```

Description
Write an OnMouseEnter event handler to take specific action when the user moves the mouse over the control. For example, you can use this event to change the associate control when the mouse is over the control, and then use the OnMouseLeave event to change it back when the mouse moves off the control.

See also: TXFUpDown.OnMouseLeave

TXFUpDown.OnMouseLeave

Occurs when the mouse pointer moves off from over the control.

```
property OnMouseLeave: TNotifyEvent;
```

Description
Write an OnMouseLeave event handler to take specific action when the user moves the mouse off the control. For example, you can use this event to undo changes that were made in an OnMouseEnter event handler.

See also: TXFUpDown.OnMouseEnter

TXFStatusBar

TXFStatusBar represents a gradient status bar.

Unit
XFSCtrls

Description
The TXFStatusBar component represents a gradient status bar. TXFStatusBar is similar to the standard TStatusBar control, but introduces several properties to control gradient style.

See also: TXFTrackBar, TXFProgressBar

TXFStatusBar.Gradient

Specifies the gradient drawing style for the status bar.

```
property Gradient: TXFGradient;
```

Description
The Gradient property holds TXFGradientBackground object that determines gradient drawing style for the status bar. The gradient drawing style IsActive for status bar only when ThemesEnabled is False.

See also: TXFTrackBar.Gradient, TXFProgressBar.Gradient

TXFStatusBar.Create

Creates and initializes a TXFStatusBar instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a gradient status bar control. Status bar added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the status bar instance. It becomes the value of the Owner property.

See also: TXFStatusBar.Destroy

TXFStatusBar.Destroy

Destroys the gradient status bar object.

```
destructor Destroy; override;
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free checks that the gradient status bar control is not nil, and only then calls Destroy.

See also: TXFStatusBar.Create

TXFTrackBar

TXFTrackBar represents a gradient track bar.

Unit
XFSCtrls

Description
The TXFTrackBar component represents a gradient track bar. TXFTrackBar is similar to the standard TTrackBar control, but introduces several properties to control gradient style.

See also: TXFStatusBar, TXFProgressBar

TXFTrackBar.Gradient

Specifies the gradient drawing style for the track bar.

```
property Gradient: TXFGradient;
```

Description
The Gradient property holds TXFGradientProgress object that determines gradient drawing style for the track bar. The gradient drawing style IsActive for track bar when ThemesEnabled is both True or False.

See also: TXFStatusBar.gradient, TXFProgressBar.Gradient

TXFTrackBar.Create

Creates and initializes a TXFTrackBar instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a gradient track bar control. Track bar added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the track bar instance. It becomes the value of the Owner property.

See also: TXFTrackBar.Destroy

TXFTrackBar.Destroy

Destroys the gradient track bar object.

```
destructor Destroy; override;
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free checks that the gradient track bar control is not nil, and only then calls Destroy.

See also: TXFTrackBar.Create

TXFProgressBar

TXFProgressBar represents a gradient progress bar.

Unit
XFSCtrls

Description
The TXFProgressBar component represents a gradient progress bar. TXFProgressBar is similar to the standard TProgressBar control, but introduces several properties to control gradient style.

See also: TXFTrackBar, TXFStatusBar

TXFProgressBar.Gradient

Specifies the gradient drawing style for the progress bar.

```
property Gradient: TXFGradient;
```

Description
The Gradient property holds TXFGradientProgress object that determines gradient drawing style for the progress bar. The gradient drawing style IsActive for progress bar only when ThemesEnabled is False.

See also: TXFStatusBar.Gradient, TXFTrackBar.Gradient

TXFProgressBar.Create

Creates and initializes a TXFProgressBar instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a gradient progress bar control. Progress bar added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the progress bar instance. It becomes the value of the Owner property.

See also: TXFProgressBar.Destroy

TXFProgressBar.Destroy

Destroys the gradient progress bar object.

```
destructor Destroy; override;
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free checks that the gradient progress bar control is not nil, and only then calls Destroy.

See also: TXFProgressBar.Create

TXDBNavigator

TXDBNavigator represents a gradient database navigator.

Unit
XDBCtrls

Description
TXDBNavigator represents a gradient database navigator. TXDBNavigator is derrived from the standard TDBNavigator control. TXDBNavigator introduces several properties to control border, gradient style and parent's background.

See also: TDBNavigator

TXDBNavigator.ApplyTheme

Determines whether the Windows Theme is used for the buttons.

```
property ApplyTheme: Boolean; {* ver. 5.1 *}
```

Description
Set ApplyTheme to true, to draw themed buttons with using Windows Themes. This property is respected only when XP Manifest is added to the application.

See also: TXDBNavigator.FineTheme, TDBNavigator.DrawingStyle

TXDBNavigator.Captions

Provides a way to customize the captions for the buttons on the database navigator.

```
property Captions: TStrings;
```

Description
Use the Captions property to supply captions of your choosing for the individual navigator buttons. Each button has a default caption. Captions allow the values of any or all of these default captions to be replaced by customized captions.

Captions is a string list. Each caption is a string. The first string in the string list becomes the caption for the first button on the navigator (the First button). The seventh hint becomes the caption for the seventh button (the Edit button).

When specifying Captions at runtime, enter an empty string for any caption that should keep the default value. Simply leave the line blank when using the string list property editor of the Object Inspector for the Captions property.

Note: To have the Captions appear at runtime, set the ShowCaptions property to true.

See also: TXDBNavigator.ShowCaptions

TXDBNavigator.DrawingStyle

Specifies the drawing style of the buttons of navigator.

```
property DrawingStyle: TButtonDrawingStyle; {* ver. 5.1 *}
```

Description
Use DrawingStyle to select drawing style for the buttons of navigator. The DrawingStyle is a master property for ApplyTheme and Gradient.Active property.

See also: TXDBNavigator.ApplyTheme, TXDBNavigator.Gradient

TXDBNavigator.Fine

Determines whether the buttons have a semi-flat or semi-3D border that provides a raised or lowered look.

```
property Fine: Boolean;
```

Description
Set Fine to true to change the raised border when the buttons are unselected and the lowered border when the button is clicked. The Fine property introduces two new borders (semi-flat or semi-3D) depend on value of Flat property. This property is respected only when the Gradient IsActive.

See also: TXDBNavigator.FineTheme, TDBNavigator.Flat, TXDBNavigator.Gradient

TXDBNavigator.FineTheme

Determines whether the buttons have a special look under Windows XP.

```
property FineTheme: TFineTheme;
```

Description
Set FineTheme to true to change the default button's theme under Windows XP. The FineTheme property introduces two new themes when the Fine property is True. When the Fine property is False or either FineTheme is ftNoChange the default button's themes leave unchanged.

See also: TXDBNavigator.Fine

TXDBNavigator.Font

Determines font for captions drawing on the buttons.

```
property Font: TFont; {* ver. 5.1 *}
```

Description
The Font property points to a TFont object that determines typographic attributes of text displayed on the buttons.

See also: TXDBNavigator.ParentFont

TXDBNavigator.Gradient

Specifies the gradient drawing style for database navigator.

```
property Gradient: TXFGradient;
```

Description
The Gradient property holds TXFGradientButton object that determines gradient drawing style for the database navigator. The gradient drawing style IsActive for navigator's buttons when ThemesEnabled is False and for navigator's panel when ParentBackground is False.

See also: TXDBNavigator.Fine

TXDBNavigator.Kind

Indicates the orientation of the navigator.

```
type
  TDBNavigatorKind = (dbnHorizontal, dbnVertical); // For C++Builder/Delphi 5,
    6, 7, 8, 2005, 2006, 2007, 2009, 2010, XE
  TDBNavigatorKind = DBCtrls.TDBNavigatorKind;     // For C++Builder/Delphi
    XE2 or higher
property Kind: TDBNavigatorKind; {* ver. 5.1 *}
```

Description
Use Kind to determine orientation of the navigator. When Kind is dbnHorizontal the XDBNavigator is oriented horizontally. When Kind is dbnVertical the XDBNavigator is oriented vertically.

See also:

TXDBNavigator.Layout

Specifies where the images appear on the bitmap buttons.

```
property Layout: TButtonLayout;
```

Description
Layout indicates whether the text appears on the left of the button (blGlyphLeft), the right of the button (blGlyphRight), the top (blGlyphTop) or the bottom (blGlyphBottom).

See also: TXDBNavigator.Captions, TXDBNavigator.Margin, TXDBNavigator.Spacing

TXDBNavigator.Margin

Specifies the number of pixels between the edge of the image and the edge of the button.

```
property Margin: Integer;
```

Description
The margin is the space between the image edge and the button edge. The format of the edges depends on the Layout of the image and text. For example, if Layout is blGlyphLeft, the margin appears between the left edge of the image and the left edge of the button. If Margin is 3, three pixels separate the image and the button edges. If Margin is 0, there is no space between the image and the button edges.

If Margin is -1 (the default value), the image and text (specified in the Captions property) are centered. The number of pixels between the image and button edge is equal to the number of pixels between the opposite edge of the button and the text.

See also: TXDBNavigator.Captions, TXDBNavigator.Layout, TXDBNavigator.Spacing

TXDBNavigator.ParentBackground

Determines whether the navigator's panel uses parent's background.

```
property ParentBackground: Boolean;
```

Description
If ParentBackground is true, the navigator's panel uses the parent's background to draw its own background. ParentBackground has effect for both ThemesEnabled is True or False.

See also: TXDBNavigator.ParentBackground

TXDBNavigator.ParentFont

Specifies where a navigator looks for its font information.

```
property ParentFont: Boolean; {* ver. 5.1 *}
```

Description
Set ParentFont to true to have a navigator which uses the same font as its parent control. If ParentFont is false, the navigator uses its own Font property. When the value of a control's Font property changes, ParentFont becomes false automatically.

See also: TXDBNavigator.Font

TXDBNavigator.ShowCaptions

Determines whether the database navigator displays captions.

```
property ShowCaptions: Boolean;
```

Description
Use ShowCaptions to determine whether Captions appear for the database navigator. To appear captions for the control the property must be true.

See also: TXDBNavigator.Captions

TXDBNavigator.Spacing

Determines where the image and text appear on a buttons.

```
property Spacing: Integer;
```

Description
Spacing determines the number of pixels between the image and the text (specified in the Captions property). The default value is 4 pixels.

If Spacing is a positive number, its value is the number of pixels between the image and text. If Spacing is 0, no pixels will be between the image and text. If Spacing is -1, the text appears centered between the image and the button edge. The number of pixels between the image and text is equal to the number of pixels between the text and the button edge opposite the glyph.

See also: TXDBNavigator.Captions, TXDBNavigator.Margin, TXDBNavigator.Layout

TXDBNavigator.Create

Creates and initializes a TXDBNavigator instance.

```
constructor Create(AOwner: TComponent); override;
```


Description
Use Create to programmatically instantiate a gradient database navigator. Navigator added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the navigator instance. It becomes the value of the Owner property.

See also: TXDBNavigator.Destroy

TXDBNavigator.Destroy

Destroys the gradient database navigator object.

```
destructor Destroy; override;
```

Description
Do not call Destroy directly in an application. Instead, call Free. Free checks that the gradient navigator control is not nil, and only then calls Destroy.

See also: TXDBNavigator.Create

TXDBNavButton

TXDBNavButton represents a gradient navigator's button.

Unit
XDBCtrls

Description
TXDBNavButton represents a gradient navigator's button. TXDBNavButton is similar to the TXFSpeedButton control, which introduces several properties to control gradient style. TXDBNavButton is a control designed for TXDBNavigator.

See also: TXFSpeedButton, TXDBNavigator

TXDBText

TXDBText represents a data-aware control that displays the value of a field on a gradient background.

Unit
XDBCtrls

Description
Use TXDBText to display the contents of a field in the current record of a dataset on a gradient background. TXDBText is derrived from the standard TDBText control, but Transparent property is default True.

See also: TXDBCheckBox

TXDBText.Transparent

Specifies whether the background of the text is transparent.

```
property Transparent: Boolean;
```

Description
Use Transparent to specify whether the background of the text is transparent. This property is **default** True.

See also: TXDBCheckBox.Transparent

TXDBText.Create

Creates and initializes a TXDBText instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a transparent data-aware text control. Data-aware text added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the text control instance. It becomes the value of the Owner property.

See also: TXDBCheckBox.Create

TXDBCheckBox

TXDBCheckBox represents a data-aware transparent check box.

Unit
XDBCtrls

Description
Use TXDBCheckBox to place a data-aware transparent check box on gradient background. TXDBCheckBox is derived from the standard TDBCheckBox control, but works effecive on gradient background.

See also: TXDBText

TXDBCheckBox.Transparent

Specifies whether the background of the check box is transparent.

```
property Transparent: Boolean;
```

Description
Use Transparent to specify whether the background of the caption is transparent. This property is **default** True and works effective on gradient background.

See also: TXDBText.Transparent

TXDBCheckBox.Create

Creates and initializes a TXDBCheckBox instance.

```
constructor Create(AOwner: TComponent); override;
```

Description
Use Create to programmatically instantiate a transparent data-aware check box. Data-aware check box added in the form designer are created automatically.

AOwner is the component that is responsible for freeing the check box control instance. It becomes the value of the Owner property.

See also: TXDBText.Create

TXDBRadioGroup

TXDBRadioGroup represents a gradient group of radio buttons connected to a database.

Unit
XDBCtrls

Description
Use TXDBRadioGroup to group a set of data-aware radio buttons on a gradient background. TXDBRadioGroup is similar to the standard TDBRadioGroup control, but introduces several properties to control gradient style and parent's background.

See also: TDBRadioGroup

