About

Auto Components Library (32-bit)

RSD software
e-mail: rsd@.tibc.tula.ru
INTERNET : HTTP:\\WWW.RSD.PP.RU
fax: +7 0872 770183

Now we distribute it in two versions:
Auto Components Library Client/Server, 25 components + Auto Form Expert - 120 USA dollars;
Auto Components Library Desktop, 13 components - 49 USA dollars.

For more information see Contacting information

It will be work properly FOREVER. We decided to remove ALL PROTECTION from our library so you can test and even use it, but it doesn't mean that Auto Components Library is free.

The components of the Auto Components Library can be divided into four groups:
Data Access components
Visual Filter ComponentsLibrary
Data Controls
Additional Components

For information about registration Auto Components Library see Registration Orders.

CONSTRUCT YOUR APPLICATION WITHOUT ANY LINE OF CODE.

Any comments and BUG reports are welcome.

See manual.doc and history.txt for more information.

Authors:

1. Andrey Telnov and Roman Eremin - idea of creating the current project (Auto Visual Library 32)
2. Andrey Telnov - programming
3. Roman Eremin and Dmitry Rogov - internal testing
4. Dmitry Rogov and Ura Sergeev - creating of the help system
5. Dmitry Rogov - creating Auto32 Expert
6. Ura Sergeev - technical support

Especial thanks for testing and suggestion to Theo Pistorius, M.D. Nazca Software (South Africa).

Registration Orders

For technical support or comments about this program, you may contact us at:
RSD software
e-mail: rsd@.tibc.tula.ru
INTERNET : HTTP:\\WWW.RSD.PP.RU
fax: +7 0872 770183

For your convenience we have contracted another company, NorthStar Solutions, to process any orders you may wish to place with your Visa, MasterCard, or Discover card. Please be sure to mention you would like to order:
Product #1418 - Auto Components Library Client/Server (auto32), $ 120.00
Product #1420 - Auto Components Library Client/Server Update(auto32), $ 90.00
Product #1466 - Auto Components Library Desktop (auto32d), $ 49.00;

INTERNET ORDERS
Visit NorthStar Solutions at
http://nstarsolutions.com/898.htm
and fill out their online order form--fast, easy and secure!

PHONED ORDERS
Calls are taken 10 am - 8 pm, EST, Monday thru Saturday.
1-800-699-6395 (From the U.S. only.)
1-803-699-6395

FAXED ORDERS
Available 24 hours.
1-803-699-5465

E-MAILED ORDERS
CompuServe: starmail
America Online: starmail
Internet: starmail@compuserve.com

MAILED ORDERS
You may register with a check or money order.
Make them payable to "NorthStar Solutions" and send them to:
PO Box 25262, Columbia, SC 29224

Please provide (or be prepared to provide) the following information:
* The program you are registering:
Product #1418 - Auto Components Library Client/Server (auto32), $ 120.00
Product #1420 - Auto Components Library Client/Server Update(auto32), $ 90.00
Product #1466 - Auto Components Library Desktop (auto32d), $ 49.00;
* Your mailing address.
* Your Visa, MasterCard, or Discover # and its expiration date (if using credit card).
* Your E-Mail address (so NorthStar Solutions can send you an E-Mail confirming your order and so We can contact you easily with any important follow-up information, upgrade announcements, etc.).

You may wish to add the above information (or a convenient link to it) to any reminder screens, web pages, etc. Note that a visually attractive registration screen can better convey your information. The MAILED ORDERS section *may* also change *if* you plan to have all non-credit card orders sent to you directly.

Data Access components:

 TAutoTable

 TAutoQuery

 TAutoFind

TMacro (object)

TMacros (object)

Visual Filter Components Library.

TAutoFilter (object)

 TCheckBoxFilter

 TComboBoxFilter

 TDateFilter

 TLookupComboFilter

 TLookupListFilter

 TEasyFilter

 TFilterLink

 TListBoxFilter

 TMaskEditFilter

 TRadioGroupFilter

 TReferenceFilter

Data Controls.

 TAutoDBGrid

 TAutoDBDateEdit

 TDBReference

 TReference

Additional Components.

 TAutoDateEdit

 TReferencePanel

The main features

Macro in Data Access Components
Visual Filter Components
Sorted TAutoDBGrid component
From TDBComboBox to TDBReference

Macro in Data Access Components

Traditionally in applying TTable and TQuery components we often have to change Filter and SQL properties in run-time which leads to error appearing and waste our time. For those error not appear so often and make this properties more convenient in applying we have created the TMacro object. That object is similar to TParam object but first of all the Filter property can also use the TMacro object and secondly this object has only two published properties: Name and Text. Before using the property where the TMacros object was declared, It copies it Text property into the place where it was used.

For example: We have the next filter property - ‘profit < &Summa’, where ‘summa’ is the macros name which has Text property assign to ‘1000’. When the component begins to apply the Filter property its value will be change to - ‘profit < 1000’.

Visual Filter Components

See also

If you use the parameters of a dynamic SQL statement or the macros in applying AutoData Access Components you have to change Value or Text property in run-time. As usual the changing Values of this properties is the result of changing some property of Delphi controls. User changes the properties of Delphi controls and your application has to Refresh DataSet with new values of TParam or TMacro properties. Using the Visual Filter Components you can create links between Macros or Params properties of the DataSet from one side and Controls from other in design time.

The Visual filter components library consists of TAutoFilter object, TFilterLink component and the set of components which have at least one property of the TAutoFilter type.

The TFilterLink component create relation between the AutoFilter object and the macro or the param object of the DataSet. On changing the Text property of the TAutoFilter, TFilterLink assign its value to the macro or param object and if DataSet is open and AutoRefresh property of TFilterLink assigns to True (it is a default value) the TFilterLink reopens or refreshes BDE filter of the DataSet.

For example: You need to create ability for users to fetch all the records from the table EMPLOYEE (Delphi DBDEMOS) where the salary is more then 'MinSalary' and user may change the value 'MinSalary' in your application by typing the necessary value .

Assign the filter property of DataSet to 'Salary>&MinSalary' or added to the SQL statement '...Salary>:MinSalary...' (or using macro '...Salary>&MinSalary...')

Drop TFilterLink and TMaskEditFilter to the form

Assign TFilterLink properties:

1. DataSet to our DataSet
2. Filter to MaskEditFilter
3. Macros or Param to 'MinSalary'

It doesn't take a lot of time, because to assign all of this properties we just need select the necessary values from the drop down lists. Now on changing text of the MaskEditFilter controls our Dataset will be refreshed.

Notes:

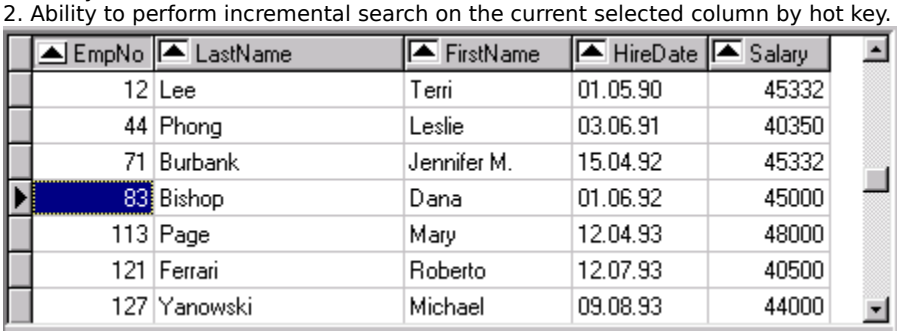
If the filter component has more then one properties of TAutoFilter type (for example TDBReference has two such properties: AutoFilter and SQLOrderFilter) you have to assign the FilterName property to the necessary Value.

If the macro, linking to the TFilterLink components, is used only in the Filter property of DataSet then the TFilterLink refreshes only BDE filter without being reopen.

See also
Data Access components
Visual Filter Components
Data Controls
Additional Components

Sorted TAutoDBGrid component

TAutoDBGrid (see the picture) is inherited from TDBGrid and has two new features:
1. Ability to sort TDBGrid columns
2. Ability to perform incremental search on the current selected column by hot key.



EmpNo	LastName	FirstName	HireDate	Salary
12	Lee	Terri	01.05.90	45332
44	Phong	Leslie	03.06.91	40350
71	Burbank	Jennifer M.	15.04.92	45332
83	Bishop	Dana	01.06.92	45000
113	Page	Mary	12.04.93	48000
121	Ferrari	Roberto	12.07.93	40500
127	Yanowski	Michael	09.08.93	44000

3. Ability to change color and font of the rows
4. Ability to view memo and graphic fields

1. Ability to sort TDBGrid columns
There are three type of the column sorted order: Ascsedent, Descendent, None
(TSortedOrder = (gsNone, gsAsc, gsDesc)). There are three possibilities to make your column sorted:

1. add the '+' (for ascsedent) or '-' (for descendent) character to the beginning of the Tcolumn DisplayText property
2. press Ctrl and Click by the left mouse button to the column Title in Design time
3. call procedure SetSortedOrder(ACol : Integer; ASortedOrder : TSortedOrder) in run-time.

The TAutoDBGrid behavior is depends on the type of linked DataSet.

1. If DataSet is Table then TAutoDBGrid allows to make columns sorted if the linked fields indexed. TAutoDBGrid sortes columns by changing the Value of the IndexName property.
2. If DataSet is TAutoQuery then you have to create the ordering macros for the SQL property and link it to the AutoFilter property of TAutoDBGrid by filter link component (TFilterLink).

For example: the SQL statement is :
'Select name, fullname, birth_date From Customer Order by &OrderField'. 'SQLOrder' is the macro name. Drop TFilterLink component to the form and assigns its properties:

a) DataSet to AutoQuery
b) Filter to AutoDBGrid
c) Macros to 'OrderField'

3. If DataSet is TQuery you have to change the SQL properties at run time on onSortChange Event.

For example:

```
procedure TForm1.AutoDBGrid1SortChange(Sender: TObject);
begin
        Query1.Close;
        Query1.SQL[Query1.SQL.Count - 1] := 'Order by '
        + AutoDBGrid1.AutoFilter.Text;
        Query1.Open;
end;
```

Note: You can't make the sorted calculated fields.

2. Ability to perform incremental search on the current selected column by hot key.
To show the find dialog just press hot key at run-time or call Find method:
For example:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  AutoDBGrid1.Find;
end;
```

Note: By default the hot key is VK_F7, but you can change it to another by assign const 'acDBGridHotKey' to your own hot key (unit aconst.pas).
3. Ability to change Color and Font of the rows.
Now you can change the Color and Fonts of the rows. Use the OnDrawFieldCellEvent event.
For example:
If the field 'profit' is less then zero then the the background color of the cell with that value would be red.

```
procedure TMainForm.AutoDBGrid1DrawFieldCellEvent(Sender: TObject;
  Field: TField; var Color: TColor; var Font: TFont);
begin
  if(Field = AutoQuery1.FindField('profit'))
  and    (AutoQuery1.FindField('profit ').AsInteger < 0) then
  Color := clRed;
```

```
end;
```
4. Ability to view Memo and Graphic fields.
Now you can view Memo or Graphic fields in AutoDBGrid. Set AutoOptions property to [adgViewImage, adgViewMemo] and resize the row heights.

From TDBLookupCombo(Box) to TDBReference

TDBReference is our first Component. Its first edition was written almost two years ago. Its third edition is included in this version.
The history of the creation of the TDBReference.
We worked under the project for one enterprise in my city in spring 1995. One of the main task of that project was to make the work with documents and dictionaries faster and more convenient. After only two weeks of testing our source we noticed that it took about forty seconds to create the simple document and more then half the time was spent on searching the necessary record in TDBLookupCombo. At that time some DataSets, already had RecordCount equal to 100 - 300 records and it took from two to five second only to fetched them, inspite of our having limited the fetched records by date and so on. On the one hand we had to decrease the number of the fetched records from the other hand the users sometimes knew the KeyValue and wanted to type it instead of searching it in the LookupCombo.
To carry out that task we decided to create our own component which would have the follow abilities:
1. type the KeyValue in EditBox and search for the necessary record in the all Table (By executing the SQL, which should fetch only one record)
2. show whether the typed Value is right or not (For example: show the firstname and lastname of the employee by typing his id)
3. show the form with DBGrids, with a strongly limited Dataset
4. add the controls to the bottom panel of the lookup form
5. allows editing on the fly
6. searching the necessary Value in the columns of the DBGrid on the fly
After the such component had been created we had killed the most of TDBLookupCombo and replace them with TDBReference. Since then we use TDBLookupCombo(Box) only if we know that the number of Records in the table will not more then 200 - 300 otherwise we apply TDBReference.

Installation for Delphi 2.x

a. If you have unregistered version

a.1. Copy the files from the \LIB.D20 to your Delphi Lib directory if you have Delphi 2.0

a.2. Copy the files from the \LIB.D21 to your Delphi Lib directory if you have Delphi 2.01

b. If you have registered version

Copy the files from the \LIB.D2 to your Delphi Lib directory

2. Run Delphi and select the OPTIONS pull-down menu. Then choose the
item INSTALL COMPONENTS...

3. Click on the ADD button. A dialog box will pop up. Click on the BROWSE button.

4. A standard file selection dialog will pop up. Select the directory that
contains the above mentioned files. In the "List Files of Type" drop-down
select *.DCU (*.PAS for registered users) files.

5. You should install (autoreg.dcu ((autoreg.pas for registered users).

Now click OK in the Install Components dialog box. Delphi's VCL will now recompiles.

Installation for Delphi 3.0

1. Run Delphi and select the COMPONENTS pull-down menu.

Then choose the item Install Packages...

2. Click on the ADD button. A dialog box will pop up. Click on the BROWSE button.

3. A standard file selection dialog will pop up. Select \LIB.D3\auto32.dpl

4. Copy the DB files from \Data directory to \Delphi 3.0\Demos\Data

Installation for C++ Builder 1.0

1. Copy the files from the \LIB.CB or (\LIB.D2 for registered users) to
your C++ Builder Lib directory \LIB
2. Run CBuilder and select the OPTIONS pull-down menu. Then choose the
item INSTALL COMPONENTS...
3. Click on the ADD button. A dialog box will pop up. Click on the BROWSE button.
4. A standard file selection dialog will pop up. Select the directory
that contains the above mentioned files. In the "List Files of Type" drop-down
select *.OBJ (*.PAS for registered users) files.
5. You should install (autoreg.obj (autoreg.pas for registered users).
Now click OK in the Install Components dialog box. VCL will now recompiles.

Installation help files.

1. Copy \help\autohelp.hlp, \help\ autohelp.kwf to \delhi\help\
2. Quit Delphi (C++ Builder) if it is running
3. Make a backup copy of \delphi\bin\delphi.hdx
4. Run the helpInst application from \delphi\help\tools
5. Open \delphi\bin\delphi.hdx
6. Select the Keywords|Add file menu option and select dbtree_v.kwf from \delphi\help
7. Select file|Save
8. Quit HelpInst
9. Run Delphi(C++ Builder). If the error 'the autohelp.hlp file not found' appears point Delphi

to the delphi\help\autohelp.hlp.
The Auto Components Library help are now merged into help system.

TAutoTable component

Properties Methods

Unit AutoDB

TAutoTable component is inherited from TTable Component.
In addition at run time, an application can supply macros values for dynamic queries and BDE filters with the Macros property and the MacroByName method.

TAutoStoredProc component (Client/Server suite only)

Properties Methods

Unit AutoDB

TAutoStoredProc component is inherited from TStoredProc Component.
In addition at run time, an application can supply macros values for dynamic queries and BDE filters with the Macros property and the MacroByName method.

Properties

property Macros
property MacrosFreeze
property MacroCount

Methods

method__MacroByName

Macros property

Applies to

TAutoQuery, TAutoTable, TAutoStoredProc

Declaration

```
property Macros [Index: Word]:TMacros;
```

When you enter a query or a filter our components creates a Macros array for the macros of a dynamic SQL statement or a BDE filter. Macros is a zero-based array of TMacro objects with an element for each macros in the query or in the filter; that is, the first parameter is Macro[0], the second Macro[1], and so on. The number of parameters is specified by MacroCount. Read-only and run-time only.
Note: Use the MacroByName method instead of Macros to avoid dependencies on the order of the macrosses.

Example

For example, suppose a TAutoTable component named AutoTable1 has the following statement for its Filter property:
'Population > &Population And continent = &continent'
An application could use Macros to specify the values of the macros as follows:

```
AutoTable1.Macros[0].Text := '10000000';
AutoTable1.Macros[1].Text := 'South America';
```

These statements would bind the macros '10000000' to the &Population macro, 'South America' to the &continent macro.

MacroCount property

Applies to

TAutoQuery, TAutoTable, TAutoStoredProc

Declaration

property MacroCount: Word;

Description

Run-time and read-only. The MacroCount property specifies how many entries the AutoTable (or the TAutoQuery) has in its Macros array, that is, how many macrosses the AutoTable (or the AutoQuery) has. Adding a new item to Macros will automatically increase the value; removing an item will automatically decrease the value.

Example

```
procedure TForm1.Button1Click(Sender: TObject);
begin
ShowMessage('The count of the AutoQuery1 macros is: ' + IntToStr(AutoQuery1.MacroCount));
end;
```

MacroByName method.

Applies to

TAutoQuery, TAutoTable, TAutoStoredProc

Declaration

function MacroByName(const Value: string): TMacro;

Description

The MacroByName method returns the element of the Macros property whose Name property matches Value. Use it to assign values to macros in a BDE filter or in a dynamic query by name.

Example

```
AutoQuery1.MacroByName('SQLOrder').Text := 'name';
```

MacrosFreeze property

Applies to

TAutoQuery, TAutoTable, TAutoStoredProc

Declaration

MacrosFreeze : TMacrosFreeze

Description

Assign MacrosFreeze to mfDesigning or mfAlways if you don't want to change Macros value by Filter components at design time or every time. (mfAlways, mfDesigning, mfNever).
mfAlways - Filter Components can always change the Text property of the Macros;
mfDesigning - Filter Components can't change the Text property of the Macros at design time;
mfNever- Filter Components can't change the Text property of the Macros.

TAutoQuery Component (Client/Server suite only)

Properties Methods

Unit AutoDB

TAutoQuery component is inherited from TQuery Component.
In addition at run time, an application can supply macros values for dynamic queries and BDE filters with the Macros property and the MacroByName method.

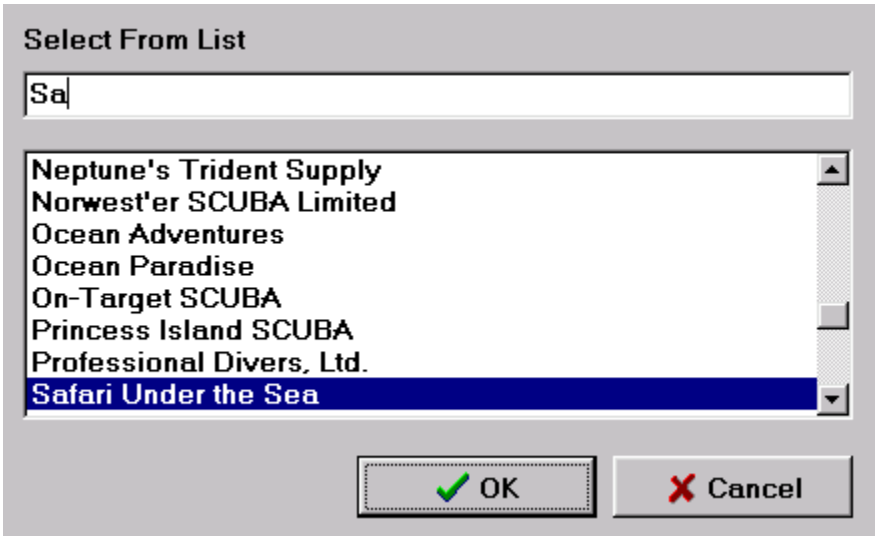
TAutoFind Component (Client/Server suite only)

Properties Methods

Unit AutoFind

Description

The TAutoFind component is an indirect descendent of TComponent. It is used to search for data of the one field in a DataBase table or query by showing the find dialog form (see the picture)
You link the TAutoFind with a DataSet when you set the value of the DataSet property and with the DataSet Field setting the value of the DataField property.
Set property Active to True to begin the search. At run-time, call method execute to begin the search. If the value from the list is chosen the DataSet cursor will be moved to the 'marked' position.



Properties

property Active;
property DataSet;
property DataField;

Methods

method Execute;

Active property.

Applies to TAutoFind

Declaration

property Active : Boolean;

Description

Set the Active property to True to begin to Search. If the DataSet, which TAutoFind is linked, is in Browse state and the property DataField is assigned right the search will begin.

Example

```
AutoFind1.DataSet := Query1;
AutoFind1.DataField := ‘country’;
AutoFind1.Active := True;
```

DataSet property.

Applies to TAutoFind

Declaration
property DataSet : TDataSet;

Description
Setting the value of the DataSet property you link the TAutoFind with a DataSet.

Example
```
AutoFind1.DataSet := Query1;
```

DataField property.

Applies to TAutoFind

Declaration

property DataField: string;

Description

The DataField property identifies the field from which the TAutoFind component displays data in its searching dialog . The field has to be owned in the DataSet which the TAutoFind component is linked.

Execute method.

Applies to TAutoFind

Declaration

procedure Execute;

Description

Call execute method to begin searching for. If the DataSet, which TAutoFind is linked, is in Browse state and the property DataField is assigned right the search will begin.

Example

```
AutoFind1.DataSet := Query1;
AutoFind1.DataField := ‘country’;
AutoFind1.Execute;
```

TMacro object

Properties Methods

unit AutoDB

Description

The TMacro object holds information about a macro of a TAutoQuery or TAutoTable. In addition to the macro text, TMacro stores the name.
You generally do not need to create a TMacro explicitly, since TAutoQuery or TAutoTable will create it as an element of its TMacros property as needed. All you have to do is assign texts to the macros by assigning its text property.
The TMacro object is a direct descendent of TObject. In addition to these properties and methods, this object also has the methods that apply to all objects.

Properties

property Name;
property Text;

Methods

method Assign;

Name property of TMacro object

Applies to TMacro

Declaration

property Name: string;

Description

The Name property is the name of the macro

Example

The following code changes the name of the &SQLOrder macro to 'Order':
 MacroByName('SQLOrder').Name := 'Order';.

Text property of TMacro object

Applies to TMacro

Declaration

property Text : String;

Description

The Text property is the Text of the macro

Example

The following code changes the text of the &SQLOrder macro:
 MacroByName('SQLOrder').Name := 'name';.

Assign method of TMacro object

Applies to TMacro

Declaration

procedure Assign(Macro: TMacro);

Description

The Assign method transfers all of the data contained in the Macro parameter to the TMacro object that calls it.

TMacros object

Properties Methods

unit AutoDB

Description

The TMacros object holds the macrosses for a TAutoQuery or TAutoTable and provides the methods to create and access those macros. Each parameter is a TMacro object.
Use the Items property to access individual macros. Call CreateMacro to create a new macros. Call AddMacro to add a new macro or RemoveMacro to take one out of the set. Call Clear to delete all macros. Use the MacroByName method to find a macro with a particular name.
The TMacros object is a direct descendent of TPersistent. In addition to these properties and methods, this object also has the methods that apply to all objects.

Properties

property Items;
property MacroText;

Methods

method Assign;
method AssignValues;
method AddMacro;
method RemoveMacro;
method CreateMacro;
method Count;
method Clear;
method MacroByNames;

Items property

Applies to TMacros

Declaration

property Items[Index: Word]: TMacro; default;

Description

Run-time only. The Items array property holds the macros (TMacro objects). Use this property when you want to work with the entire set. Index identifies the index in the range 0..Count - 1. While you can use Items to reference a particular macro by its index, the MacroByName method is recommended to avoid depending on the order of the parameters.

Example

AutoQuery1.Macros.Items[0].Text := ‘name’;
(Or AutoQuery1.Macros[0].Text := ‘name’; is the same)

MacroText property

Applies to TMacros

Declaration

property MacroText[const MacroName: string]: String;

Description

Run-time only. Use this property when you want to assign or get the Text property of the Macro. MacroName is the Name of the Macro.

Example

```
AutoQuery1.Macros.MacroText['SQLOrder'] :=   'name';
```

AddMacro method

Applies to TMacro

Declaration

procedure AddMacro(Value: TMacro);

Description

AddMacro adds Value as a new parameter to the Items property.

Assign method

Applies to TMacros

Declaration

procedure Assign(Source: TPersistent);

Description

If Source is another TMacros object, Assign discards any current parameter information and replaces it with the information from Source. Use this method to save and restore a set of macros information or copy another object's information.

Example

```
var SavedMacros: TMacros;
...
{ Initialize SavedMacros }
SavedMacros := TMacros.Create;
try
   { Save the macros for AutoTable1 }
   SavedMacros.Assign(AutoTable1. Macros);
{ Do something with AutoTable1}
...
{ Restore the macros to AutoTable1}
AutoTable1. Macros.Assign(SavedMacros);
finally
   SavedMacros.Free;
end;
```

AssignValues method

Applies to TMacros

Declaration

procedure AssignValues(Value: TMacros);

Description

For each entry in Items, the AssignValues method attempts to find a macros with the same Name property in Value. The Text property of Macro from the Value parameter is assigned to the Items entry. Entries in Items for which no match is found are left unchanged.

Example

AutoQuery2.Macros.Assign(AutoQuery1. Macros);

Clear method

Applies to TMacros

Declaration
procedure Clear;

Description
The Clear method deletes all macros information from Items.

Example
AutoQuery2.Macros.Clear;

Count method

Applies to TMacros

Declaration
function Count: Integer;

Description
The Count method returns the number of entries in Items.

Example
```
for I := 0 to AutoTable1.Macrosses.Count - 1 do
 AutoTable1.Macross[I].Text := IntToStr(I);
```

CreateMacro method

Applies to TMacros

Declaration

function CreateMacro(const MacroName: string): TMacro;

Description

The CreateMacro method attempts to create a new entry in Items, using the MacroName parameter.

MacroByName method

Applies to TMacros

Declaration

function MacroByName(const Value: string): TMacro;

Description

The MacroByName method finds a macros with the name passed in Value. If a match is found, MacroByName returns the macro. Otherwise, returns Nil. Use this method rather than a direct reference to the Items property if you need to get a specific macro to avoid depending on the order of the entries.

Example

The following code changes the text of the &SQLOrder macro:
MacroByName('SQLOrder').Name := 'name';.

RemoveMacro method

Applies to TMacros

Declaration

procedure RemoveMacro(Value: TMacro);

Description

RemoveMacro removes Value from the Items property.

TAutoFilter object

Properties Methods Events

Unit afilter

Description

TAutoFilter object is the basic object in Visual Filter Component Library. This object is used to create the Filter Components.
The Owner property identifies the owner component of the object. This component has to assign the Value properties and can change the Name property. The Name property is not stored. Its default value is 'Auto'. If the AutoFilter is linked with a param (or a macros) of a DataSet, then a value of a param property (or a Text of a macros property) is assigned to the Text property of the AutoFilter. The Text property is equal to TextBefore + Value + TextAfter.
Link the autofilter object with a DataSet by specifying the filter link component (TFilterLink). You have to determine at least the DataSet, the Filter and the Param (or the Macros) properties of the TFilterLink component.
The TAutoFilter object is a direct descendent of TPersistent. In addition to these properties and methods, this object also has the methods that apply to all objects.

See also

TFilterLink

Properties

property Name;
property Text;
property TextAfter;
property TextBefore;
property Value;

Methods

method Create;
method Destroy;

Events

event OnBeforeChange;
event OnAfterChange;

Name property

Applies to TAutoFilter

Declaration

Name : String;

Description

The Name property contains the name of the AutoFilter as referenced by the filter link components. By default, It assign to 'Auto'. You may change it to suit your needs.

Example

```
AutoFilter.Name := 'SQLOrder';
```

Text property

Applies to TAutoFilter

Declaration

property Text : String;

Description

If the AutoFilter is linked with a param (or a macros) of a DataSet, then a value of a param property (or a Text of a macros property) is assigned to the Text property of the AutoFilter. The Text property is equal to TextBefore + Value + TextAfter. You never have to change the Text property directly.

TextAfter property

Applies to TAutoFilter

Declaration

property TextAfter : String;

Description

In changing the TextAfter property changes the Text property is also changed. The Text property is equal to TextBefore + Value + TextAfter.

Example

MaskEditFilter1.AutoFilter.TextAfter := ‘ Desc’;

TextBefore property

Applies to TAutoFilter

Declaration

property TextBefore : String;

Description

In changing the TextBefore property changes the Text property is also changed. The Text property is equal to TextBefore + Value + TextAfter.

Example

MaskEditFilter1.AutoFilter.TextBefore := ‘Order by ’;

Value property

Applies to TAutoFilter

Declaration

property Value: String;

Description

In changing the Value property changes the Text property is also changed. The Text property is equal to TextBefore + Value + TextAfter.

Example

MaskEditFilter1.AutoFilter.Value := ‘last_name’;

Create method

Applies to TAutoFilter

Declaration

constructor Create(AOwner : TComponent);

Description

The Create method constructs, initializes a new autofilter object and assigns its Owner property to AOwner

Example

FAutoFilter := TAutoFilter.Create(self);

Destroy method

Applies to TAutoFilter

Declaration

destructor Destroy;

Description

The Destroy method destroys and disposes of an autofilter object instance.

Example

```
if(FAutoFilter <> Nil) then
   FAutoFilter.Destroy;
```

OnAfterChange event

Applies to TAutoFilter

Declaration

OnAfterChange: TNotifyEvent;

Description

The OnAfterChange event specifies which event handler should execute when the Text property has been changed.

OnBeforeChange event

Applies to TAutoFilter

Declaration

OnBeforeChange: TNotifyEvent;

Description

The OnBeforeChange event specifies which event handler should execute before the Text property is changed.

TCheckBoxFilter component (Client/Server suite only)

Properties Events

unit filtcomp

Description

The TCheckBoxFilter component is a direct descendent of TCheckBox.
The ValueChecked property assigns itself to the Value of the AutoFilter property if the Checked property is True.
The ValueUnChecked property assigns itself to the Value of the AutoFilter property if the Checked property is False.
Link the autofilter object with a DataSet by specifying the filter link component (TFilterLink). You have to determine at least the DataSet, the Filter and the Param (or the Macros) properties of the TFilterLink component.

Properties

property AutoFilter
property ValueChecked
property ValueUnChecked

Events

event OnAfterFilterChange
event OnBeforeFilterChange

AutoFilter property

Applies to TCheckBoxFilter,TReferenceFilter, TRadioGroupFilter, TComboBoxFilter, TDateFilter, TLookupComboFilter, TLookupListFilter, TEasyFilter, TFilterLink, TListBoxFilter, TMaskEditFilter, TAutoDBGrid

Declaration

property AutoFilter : TAutoFilter

Description

The AutoFilter property is the instance of the owned auto filter object.

Example

CheckBoxFilter1.AutoFilter.TextBefore := 'Order by ';

ValueChecked property

Applies to TCheckBoxFilter

Declaration

property ValueChecked : String;

Description

The ValueChecked property assigns itself to the Value of the AutoFilter property if the Checked property is True

Example

CheckBoxFilter1.ValueChecked := ‘T’;

ValueUnChecked property

Applies to TCheckBoxFilter

Declaration

property ValueUnChecked : String;

Description

The ValueUnChecked property assigns itself to the Value of the AutoFilter property if the Checked property is False.

Example

CheckBoxFilter1.ValueUnChecked := ‘F’;

OnAfterFilterChange event

Applies to TCheckBoxFilter,TReferenceFilter, TRadioGroupFilter, TComboBoxFilter, TDateFilter, TLookupComboFilter, TLookupListFilter, TEasyFilter, TListBoxFilter, TMaskEditFilter

Declaration

OnAfterFilterChange: TNotifyEvent;

Description

The OnAfterFilterChange event specifies which event handler should execute when the Text property of the owned auto filter object has been changed.

Example

```
procedure TForm1.CheckBoxFilter1AfterFilterChange(Sender: TObject);
begin
    ShowMessage('You have change the checked property of the   CheckBoxFilter1');
end;
```

OnBeforeFilterChange event

Applies to TCheckBoxFilter,TReferenceFilter, TRadioGroupFilter, TComboBoxFilter, TDateFilter, TLookupComboFilter, TLookupListFilter, TEasyFilter, TListBoxFilter, TMaskEditFilter

Declaration

OnBeforeFilterChange: TNotifyEvent;

Description

The OnBeforeFilterChange event specifies which event handler should execute before the Text property of the owned auto filter object is changed.

Example

```
procedure TForm1.CheckBoxFilter1BeforeFilterChange(Sender: TObject);
begin
  ShowMessage('You have change the checked property of the   CheckBoxFilter1');
end;
```

TComboBoxFilter component (Client/Server suite only)

Properties Events

unit filtcomp

Description

The TComboBoxFilter component is a direct descendent of TComboBox.
The selected item in the combo box assigns its Value to the Value of the AutoFilter property. If you want to assign to the Value of the AutoFilter property the different value from the Value of Items you have to assign to the Item the string with such pattern : '<Display text> , <Assigned Value>'.
Link the auto filter object with a DataSet by specifying the filter link component (TFilterLink). You have to determine at least the DataSet, the Filter and the Param (or the Macros) properties of the TFilterLink component.

Properties

property AutoFilter
property Items

Events

event OnAfterFilterChange
event OnBeforeFilterChange

Items property

Applies to TComboBoxFilter, TLookupComboFilter, TLookupListFilter, TListBoxFilter,TRadioGroupFilter

Declaration

property Items: TStrings;

Description

The selected item in the combo box assign its Value to the Value of the AutoFilter property. If you want to assign to the Value of the AutoFilter property the different value from the Display value of Items you have to assign to the Item the string with such pattern : '<Display text> , <Assigned Value>'.

Example

Assign the array of Item the such texts:
ComboBoxFilter1.Items.Add('All countries', 0);
ComboBoxFilter1.Items.Add('The population is more then 5000000', 5000000);
ComboBoxFilter1.Items.Add('The population is more then 50000000', 50000000);
ComboBoxFilter1.Items.Add('The population is more then 100000000', 100000000);
The Value of the auto filter object will be assign to: '0', '5000000', '50000000', '100000000'.

TDateFilter component

Properties Events

unit filtcomp

Description

The TDateFilter component is a direct descendent of TAutoDateEdit.
The Date property assign itself to the Value of the AutoFilter property.
Link the auto filter object with a DataSet by specifying the filter link component (TFilterLink). You have to determine at least the DataSet, the Filter and the Param (or the Macros) properties of the TFilterLink component.

Properties

property AutoFilter

Events

event OnAfterFilterChange
event OnBeforeFilterChange

TLookupComboFilter component

Properties Events

unit filtcomp

Description

TLookupComboFilter lets you provide the user with a convenient drop-down list filter of lookup items that require data from DataSet.
Set KeyField to the field you want copied into Value property AutoFilter object. Set ListField to display a field other than KeyField in the combo box.
Set items property if you want to display and assign the strings which there are not in the DataSet. Items always appear at the top of the dropdown listbox and they are not a scrollable. If you want to assign to the Value of the AutoFilter property the different value from the Value of Items you have to assign to the Item the string with such pattern : '<Display text> ,<Assigned Value>'.
The selected item in the lookup list box assign its Value to the Value of the AutoFilter property.
Link the auto filter object with a DataSet by specifying the filter link component (TFilterLink). You have to determine at least the DataSet, the Filter and the Param (or the Macros) properties of the TFilterLink component.
In addition to these properties and methods, this component also has the properties and methods that apply to all components.

Properties

property AutoFilter
property Items
property ItemsColor
property ItemsAligment

Events

event OnAfterFilterChange
event OnBeforeFilterChange

property ItemsColor

Applies to TLookupComboFilter, TLookupListFilter

Declaration

ItemsColor : TColor

Description

The ItemsColor is the background on which the Items are appear. Default Value of the ItemsColor is clBtnFace.

property ItemsAligment

Applies to TLookupComboFilter, TLookupListFilter

Declaration

ItemsAligment : TAligment

Description

The ItemsAlignment property specifies how text is aligned within the component.
These are the possible values:

Value	Meaning
taLeftJustify	Align text to the left side of the control
taCenter	Center text horizontally in the control
taRightJustify	Align text to the right side of the control

TLookupListFilter component (Client/Server suite only)

Properties Events

unit filtcomp

Description

TLookupListFilter lets you provide the user with a convenient list box filter of lookup items that require data from DataSet.
Set KeyField to the field you want copied into Value property AutoFilter object. Set ListField to display a field other than KeyField in the combo box.
Set items property if you want to display and assign the strings which there are not in the DataSet. Items always appear at the top of the dropdown listbox and they are not a scrollable. If you want to assign to the Value of the AutoFilter property the different value from the Value of Items you have to assign to the Item the string with such pattern : '<Display text> ,<Assigned Value>'.
The selected item in the lookup list box assign its Value to the Value of the AutoFilter property.
Link the auto filter object with a DataSet by specifying the filter link component (TFilterLink). You have to determine at least the DataSet, the Filter and the Param (or the Macros) properties of the TFilterLink component.
In addition to these properties and methods, this component also has the properties and methods that apply to all components.

Properties

property AutoFilter
property Items
property ItemsColor
property ItemsAligment

Events

event OnAfterFilterChange
event OnBeforeFilterChange

TEasyFilter component (Client/Server suite only)

Properties Events

unit filtcomp

Description

The TEasyFilter component is a direct descendent of TComponent.
The Value property assign itself to the Value of the AutoFilter property.
Link the auto filter object with a DataSet by specifying the filter link component (TFilterLink). You have to determine at least the DataSet, the Filter and the Param (or the Macros) properties of the TFilterLink component.

Properties

property AutoFilter
property Value;

Events

event OnAfterFilterChange
event OnBeforeFilterChange

Value property

Applies to TEasyFilter

Declaration

property Value : String;

Description

The Value property assign itself to the Value of the AutoFilter property.

Example

CheckBoxFilter1.Value := ‘1000’;

TFilterLink component

Properties Methods

unit afilter

Description

The TFilterLink component is a direct descendent of TComponent.
The TFilterLink is the auto filter link component.
The filter link component links the auto filter object with a DataSet. You have to determine at least the DataSet, the Filter and the Param (or the Macros) properties of the TFilterLink component. If the filter component has more the one auto filter object the FilterName property also has to be assigned.
If the AutoRefresh property is True on Changing the Text property of the linked auto filter the linked Dataset will be reopened or its BDE filter will be refreshed.

Properties

property AutoFilter
property AutoRefresh
property DataSet
property Filter
property FilterName
property Macros
property Param

Methods

method RefreshDataSet

AutoRefresh property

Applies to TFilterLink

Declaration

property AutoRefresh : Boolean;

Description

If the AutoRefresh property is True on Changing the Text property of the linked auto filter the linked Dataset will be reopen or its BDE filter will be refreshed.

Example

FilterLink1.AutoRefresh := True;

DataSet property

Applies to TFilterLink

Declaration

property DataSet : TDataSet;

Description

The DataSet property is the instance of the linked DataSet component.

Example

```
ShowMessage('The filter ' + FilterLink1.AutoFilter.Name +
        ' is linked with DataSet ' + FilterLink1.DataSet.Name);
```

Filter property

Applies to TFilterLink

Declaration

property Filter: TComponent;

Description

The Filter property is the instance of the linked filter component.

Example

```
FilterLink1.Filter := EasyFilter1;
```

FilterName property

Applies to TFilterLink

Declaration

property FilterName : String;

Description

If the filter component has more the one auto filter object the FilterName property has been assigned.

Example

```
FilterLink1.Filter := ReferenceFilter1;
FilterLink1.FilterName := 'SQLOrder';
```

Macro property

Applies to TFilterLink

Declaration

property Macro : String;

Description

The Macro property is name of the macro object. The Text property of this macros object is assigned to the Text property of the linked autofilter object.

Example

FilterLink1.Macro := AutoTable1.Macros[0].Name;

Param property

Applies to TFilterLink

Declaration

property Param: String;

Description

The Param property is name of the param object. The value property of this Param object is assigned to the Text property of the linked autofilter object.

Example

FilterLink1.Param := AutoTable1.Param[0].Name;

RefreshDataSet method

Applies to TFilterLink

Declaration
procedure RefreshDataSet;

Description
RefreshDataSet method reopens or refreshes BDE filter linked DataSet.

Example
FilterLink2.RefreshDataSet;

TListBoxFilter component (Client/Server suite only)

Properties Events

unit filtcomp

Description

The TListBoxFilter component is a direct descendent of TListComboBox.
The selected item in the combo box assign its Value to the Value of the AutoFilter property. If you want to assign to the Value of the AutoFilter property the different value from the Value of Items you have to assign to the Item the string with such pattern : '<Display text> , <Assigned Value>'.
Link the auto filter object with a DataSet by specifying the filter link component (TFilterLink). You have to determine at least the DataSet, the Filter and the Param (or the Macros) properties of the TFilterLink component.

Properties

property AutoFilter
property Items

Events

event OnAfterFilterChange
event OnBeforeFilterChange

 **TMaskEditFilter component (Client/Server suite only)**

Properties Events

unit filtcomp

Description

The TMaskEditFilter component is a direct descendent of TMaskEditFilter.
The Text property assign itself to the Value of the AutoFilter property.
Link the auto filter object with a DataSet by specifying the filter link component (TFilterLink). You have to determine at least the DataSet, the Filter and the Param (or the Macros) properties of the TFilterLink component.

Properties

property AutoFilter

Events

event OnAfterFilterChange
event OnBeforeFilterChange

TRadioGroupFilter component (Client/Server suite only)

Properties Events

unit filtcomp

Description

The TRadioGroupFilter component is a direct descendent of TRadioGroup.
The selected item in the combo box assign its Value to the Value of the AutoFilter property. If you want to assign to the Value of the AutoFilter property the different value from the Value of Items you have to assign to the Item the string with such pattern : '<Display text> , <Assigned Value>'.
Link the auto filter object with a DataSet by specifying the filter link component (TFilterLink). You have to determine at least the DataSet, the Filter and the Param (or the Macros) properties of the TFilterLink component.

Properties

property AutoFilter
property Items

Events

event OnAfterFilterChange
event OnBeforeFilterChange

TReferenceFilter component

Properties Events

unit filtcomp

Description

The TReferenceFilter component is a direct descendent of TReference.
If ReferenceState is rsInvalided than ErrorValue property assigns to the Value of the AutoFilter property.
If ReferenceState is rsNulled than NullValue property assigns to the Value of the AutoFilter property.
If ReferenceState is rsValided then the Value of the AutoFilter property is assigned by the Value of the Field specifying the AssignField property. If AssignField is not defined the AutoFilter property is assigned by the Value of the Field specifying the KeyField property.
Link the auto filter object with a DataSet by specifying the filter link component (TFilterLink). You have to determine at least the DataSet, the Filter and the Param (or the Macros) properties of the TFilterLink component.

Properties

property AssignField
property AutoFilter

Events

event OnAfterFilterChange
event OnBeforeFilterChange

AssignField property

Applies to TReferenceFilter

Declaration

AssignField : String;

Description

If ReferenceState is rsValided then the Value of the AutoFilter property is assigned by the Value of the Field specifying the AssignField property. If AssignField doesn't define the AutoFilter property is assigned by the Value of the Field specifying the KeyField property.

Example

ReferenceFilter1.AssignFiled := 'employee_id';

TAutoDBGrid component

Properties Methods Events

unit adbgrid

Description

The TAutoDBGrid component is a direct descendent of TDBGrid and has two new features:
1. Ability to sort TDBGrid columns
2. Ability to execute search of the current selected column by hot key.
3. Ability to change color and font of the rows
4. Ability to view memo and graphic fields

1. Ability to sort TDBGrid columns
There are three type of the column sorted order: Ascsedent, Descendent, None
(TSortedOrder = (gsNone, gsAsc, gsDesc)). There are three possibilities to make your column sorted:

1. add the '+' (for ascsedent) or '-' (for descendent) character to the beginning of the Tcolumn DisplayText property
2. press Ctrl and Click by the left mouse button to the column Title in Design time
3. call procedure SetSortedOrder(ACol : Integer; ASortedOrder : TSortedOrder) in run-time.

The TAutoDBGrid behavior is depends on the type of linked DataSet.

1. If DataSet is Table then TAutoDBGrid allows to make columns sorted if the linked fields indexed. TAutoDBGrid sortes columns by changing the Value of the IndexName property.
2. If DataSet is TAutoQuery then you have to create the ordering macros for the SQL property and link it to the AutoFilter property of TAutoDBGrid by filter link component (TFilterLink).

For example: the SQL statement is :
'Select name, fullname, birth_date From Customer Order by &OrderField'. 'SQLOrder' is the macro name. Drop TFilterLink component to the form and assigns its properties:

a) DataSet to AutoQuery
b) Filter to AutoDBGrid
c) Macros to 'OrderField'

3. If DataSet is TQuery you have to change the SQL properties at run time on onSortChange Event.

For example:

```
procedure TForm1.AutoDBGrid1SortChange(Sender: TObject);
begin
        Query1.Close;
        Query1.SQL[Query1.SQL.Count - 1] := 'Order by '
        + AutoDBGrid1.AutoFilter.Text;
        Query1.Open;
end;
```

Note: You can't make the sorted calculated fields.

2. Ability to perform incremental search on the current selected column by hot key.
To show the find dialog just press hot key at run-time or call Find method:
For example:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  AutoDBGrid1.Find;
end;
```

Note: By default the hot key is VK_F7, but you can change it to another by assign const 'acDBGridHotKey' to your own hot key (unit aconst.pas).
3. Ability to change Color and Font of the rows.
Now you can change the Color and Fonts of the rows. Use the OnDrawFieldCellEvent event.
For example:
If the field 'profit' is less then zero then the the background color of the cell with that value would be red.

```
procedure TMainForm.AutoDBGrid1DrawFieldCellEvent(Sender: TObject;
  Field: TField; var Color: TColor; var Font: TFont);
begin
  if(Field = AutoQuery1.FindField('profit'))
  and    (AutoQuery1.FindField('profit ').AsInteger < 0) then
  Color := clRed;
end;
```

4. Ability to view Memo and Graphic fields.
Now you can view Memo or Graphic fields in AutoDBGrid. Set AutoOptions property to [adgViewImage, adgViewMemo] and resize the row heights.

Properties

property AutoFilter
property CurrentSortColumn

Methods

Events

event OnSortChange

CurrentSortColumn property

Applies to TAutoDBGrid

Declaration

property CurrentSortColumn : smallest;

Description

The CurrentSortColumn property is the number of the column which is currently sorted. If the AutoDBGrid is not sorted the CurrentSortColumn is -1.

Example

```
if(AutoDBGrid1.CurrentSortColumn > 0) then
  ShowMessage('AutoDBGrid1 is sorted by ' + AutoDBGrid1.Columns[AutoDBGrid1.CurrentSortColumn].FieldName);
```

Find method

Applies to TAutoDBGrid

Declaration
procedure Find;

Description
Call method Find to show the find dialog with the list of Values of the selected Column.

GetSortedOrder method

Applies to TAutoDBGrid

Declaration

function GetSortedOrder(ACol : Integer) : TSortedOrder;

Description

Method GetSortedOrder return the type of the column sorted order. ACol is the number of the column. (TSortedOrder = (gsNone, gsAsc, gsDesc)).

SetSortedOrder method

Applies to TAutoDBGrid

Declaration

procedure SetSortedOrder(ACol : Integer; ASortedOrder : TSortedOrder);

Description

Method SetSortedOrder set the new type of the column sorted order. ACol is the number of the column.
ASortedOrder is the new sortedorder type.
(TSortedOrder = (gsNone, gsAsc, gsDesc)).

OnSortChange event

Applies to TAutoDBGrid

Declaration
property OnSortChange : TNotifyEvent;

Description

The OnSortChange event specifies which event handler should execute when the CurrentSortCoulumn property has been changed.

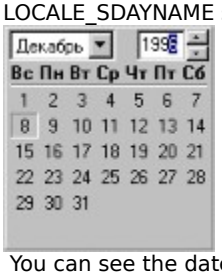
TAutoDBDateEdit component

Properties

Unit aDatEdit

Description

The TAutoDBDateEdit component is a direct descendent of TDBEdit.
The TAutoDBDateEdit control has the bitbutton. On the onclick event of the bitbutton the date setting dialog form will be shown. The name of the months and days are fetched from the LOCALE_SMONTHNAME and LOCALE_SDAYNAME system locale entries.



You can see the date setting dialog form for Russian Windows 95.
The Date property is the date showing on the edit box.
If the IsDateNow property is True and the Date property is Nil then the current date is assigned to the Date property.

Properties

property Date
property IsDateNow

Date property

Applies to TAutoDBDateEdit , TAutoDateEdit

Declaration
property Date: TDateTime;

Description
The Date property is the date showing on the edit box.

IsDateNow property

Applies to TAutoDBDateEdit,TAutoDateEdit

Declaration

property IsDateNow : Boolean;

Description

If the IsDateNow property is True and the Date property is Nil then the current date is assigned to the Date property.

TDBReference component

Properties Methods Events

unit adbgrid

Description

The TDBReference component is a direct descendent of TReference.
The TReference is inherited from TWinControl.
The TDBReference is more powerful then TDBLookupComboBox.
The TDBReference component has the follow abilities:

1.type the KeyValue in EditBox and search for the necessary record in the all Table (By executing the
2.SQL, which should fetch only one record)
3.show whether the typed Value is right or not (For example: show the firstname and lastname of the employee by typing his id)
4.show the form with DBGrids, with a strongly limited Dataset
5.add the controls to the bottom panel of the lookup form
6.editing on the fly
7. perform searching the necessary Value in the columns of the DBGrid on the fly

Set the DataSource property to the data source that will receive the user's selection. Set the DataField to either the field to receive the user's selection or a lookup field.
Set the ListSource property to the data source of the table holding the lookup items.
Set KeyField to the field you want to type in the Edit Box. Set AssingField to the field you want it to be copied into DataField. If AssingField is Nil then KeyField will be copied into DataField. Set Columns to display fields in the popup form.
If ReferenceState is rsInvalided than ErrorValue property will be copied into DataField.
If ReferenceState is rsNulled than NullValue property will be copied into DataField.
Set the VisibleText property to True to show the Reference Text. Set PatternText to display text of the Reference Text. To display the value of the Field use '&' character ('&fieldname').
Set the ReferencePanel property to display the ReferencePanel control on the bottom of the PopUp Form.
Set the UseQuery property to True to execute a query on changing the reference Edit Box. The SQL statement is automatically created. This Query try to fetch only one record searching for the entered Text in the Table (the TableName property is name of that Table) in AssignField or KeyField field. It's very useful for the remote DataBase especially if there are a lot of records in the table. If the UseQuery property is False the searching is executed in the DataSet linking with ListSource.

Properties

AlignText
AssignField
CanUseQuery
Columns
CurrentSortColumn
DataField
DataSource
EditWidth
ErrorValue
Font
FontText
Glyph
KeyField
ListSource
NumGlyphs
Options
PatternText
ReferencePanel
State
SQLOrderFilter
TableName
Text
TextOnError
UseQuery
VisibleText
WinHeight
WinWidth

Methods

ChangeTextByPattern
FindByKeyField
FindByKeyFieldInList
GetQuery
GotoKeyFieldValue
RefreshText

Events

OnChange

AlignText property

Applies to TDBReference

Declaration

property AlignText : TAlignReferenceText;

Description

TAlignReferenceText = (rtLeft, rtRight);
The AlignText property determines where the reference edit is allocated (rtLeft is the default Value)

Example

DDreference1.AlignText := rtRight;

AssignField property

Applies to TDBReference

Declaration

property AssignField : String

Description

The AssingField property is the field which you want it to be copied into DataField. If AssingField is Nil then KeyField will be copied into DataField.

Example

DDreference1.AssignField := ‘id’

CanUseQuery property

Applies to TDBReference

Declaration

CanUseQuery : Boolean;

Description

Run-time and read only. If the UseQuery property is True and the Query is executed without errors the CanUseQuery property assigns True otherwise assigns False and the searching makes in the DataSet linking with the ListSource property.

Columns property

Applies to TDBReference

Declaration
property Columns : TDBGridColumns

Description
The Columns property is the columns of the AutoDBGrid which use in the pop up form.

CurrentSortColumn property

Applies to TDBReference

Declaration

property CurrentSortColumn : integer

Description

The CurrentSortColumn property is the CurrentSortColumn property of the AutoDBGrid which use in the pop up form.

Example

```
DBReference1.CurrentSortColumn := 2;
```

DataField property

Applies to TDBReference

Declaration
property DataField: string;

Description
The DataField property identifies the field from which the reference edit box.

DataSource property

Applies to TDBReference

Declaration

property DataSource: TDataSource;

Description

The DataSource property determines where the component obtains the data to display. Specify the data source component that identifies the DataSet the data is found in.

EditWidth property

Applies to TDBReference

Declaration
property EditWidth : Integer

Description
The EditWidth property is the width of the reference edit box

Example
DBReference1.EditWidth := 75;

ErrorValue property

Applies to TDBReference

Declaration
property ErrorValue : String

Description
If ReferenceState is rsInvalided then ErrorValue property will be copied into DataField.

Example
DBReference1.ErrorValue := ‘unknown’;

Font property

Applies to TDBReference

Declaration
property Font: TFont;

Description
The Font property is a font object of the reference edit box.

FontText property

Applies to TDBReference

Declaration
property FontText: TFont;

Description
The FontText property is a font object of the reference Text.

Glyph property

Applies to TDBReference

Declaration
property Glyph: TBitmap;

Description
The Glyph property is a glyph object of the reference BitButton.

KeyField property

Applies to TDBReference

Declaration

property KeyField : String

Description

The KeyField property is the field which you want to type in the Edit Box. If AssingField is Nil then KeyField will be copied into DataField.

Example

DBReference.KeyField := 'last_name'

ListSource property

Applies to TDBReference

Declaration
property ListSource : TDataSource

NumGlyphs property

Applies to TDBReference

Declaration
property NumGlyphs: Integer;

Description
The NumGlyphs property is a numglyphs property of the reference BitButton.

Options property

Applies to TDBReference

Declaration
property Options : TDBGridOptions

Description
The Options property is the Options property of the AutoDBGrid which use in the pop up form.

PatternText property

Applies to TDBReference

Declaration

property PatternText : String

Description

The PatternText property is the pattern of the reference text. To display the value of the Field use ‘&’ character (‘&fieldname’). The PatternText property applies if the State is rsValided.
See also TextOnError.

Example

DBReference1.ParttenText := ‘&capital is the capital of &country’
‘capital’ and ‘country’ is the field name.

ReferencePanel property

Applies to TDBReference

Declaration

property ReferencePanel : TReferencePanel

Description

The ReferencePanel property is the instance to the ReferencePanel control. The ReferencePanel control display on the bottom of the PopUp form. You can throw Delphi or Filter controls on it and they appear on the Popup form.

Example

```
DBReference1.ReferencePanel := ReferencePanel1;
```

SQLOrderFilter property

Applies to TDBReference

Declaration
property SQLOrderFilter: TAutoFilter

Description
The SQLOrderFilter property is the instance of the auto filter object of the AutoDBGrid which use in the pop up form.

State property

Applies to TDBReference

Declaration

State : TReferenceState;

Description

Run-time and read only. The State property specifies the current state of the reference. The possible values are those of the TReferenceState type:

- rsInvalided when the text entered into the reference edit box is invalidated
- rsNulled when there is no text in the reference edit box
- rsValided when the text entered into the reference edit box is typed right

TableName property

Applies to TDBReference

Declaration

property TableName : TFileName

Description

The TableName property is used if the UseQuery property is True. The reference query will try to fetch the record from the table which name is TableName.

Example

DBReference1.TableName := ‘EMPLOYEE’;

Text property

Applies to TDBReference

Declaration
property Text : String

Description
The Text property determines the text that appears within an reference edit box control.

Example
DBReference1.Text := ‘num_123’;

TextOnError property

Applies to TDBReference

Declaration

property TextOnError : String

Description

The TextOnError property display its text in the reference text. The TextOnError property applies if the State is rsInvalided. The default Value is ‘Invalidate’
See also PatternText.

Example

DBReference1.TextOnError := ‘There is not the employee with the such KeyValue’;

UseQuery property

Applies to TDBReference

Declaration

property UseQuery : Boolean

Description

If the UseQuery property is True then on changing the reference Text, instead of searching for the entered Text in the DataSet linked with the ListSource, the query is executed to search the record in the hole Table. The SQL statement is automatically created. This Query try to fetch only one record searching for the entered Text in the Table (the TableName property is name of that Table) in AssignField or the KeyField field. It's very useful for the remote DataBase especially if there are a lot of records in the table. If the UseQuery property is False the searching is made in the DataSet linking with ListSource. If the Query is executed without errors the CanUseQuery property is assigned True otherwise it is assigned False and the search is made in the DataSet.

Example

```
DBReference1.UseQuery := True;
```

VisibleText property

Applies to TDBReference

Declaration
property VisibleText : Boolean

Description
If the VisibleText property is True the reference Text displays.

Example
DBReference1.VisibleText := True;

WinHeight property

Applies to TDBReference

Declaration

property WinHeight : Integer

Description

The WinHeight property is the Height of the PopUp Form. You can change its Value in the Component Editor in design time.

Example

```
DBReference1.WinHeight := 200;
```

WinWidth property

Applies to TDBReference

Declaration

property WinWidth : Integer

Description

The WinWidth property is the Width of the PopUp Form. You can change its Value in the Component Editor in design time.

Example

DBReference1. WinWidth := 300;

ChangeTextByPattern method

Applies to TDBReference

Declaration

function ChangeTextByPattern(St : String) : String

Description

If the State property is Validated the result is the text where the St parameter is your Pattern Text

Example

St := ChangeTextByPattern('&capital is capital of &country');
Where the &capital and &country is the name of the fields. The possible result is:
Moscow is capital of Russia.

GetQuery method

Applies to TDBReference

Declaration

function GetQuery : TQuery

Description

Return the reference query. If the UseQuery property is False the result will be Nil.

Example

```
if(DBReference1.GetQuery <> Nil) And Not (DBReference1.GetQuery.EOF) then begin
   ShowMessage('The Value of the KeyField of the reference query is: ' +
DBReference1.GetQuery.FindField('KeyField').AsString);
end;
```

FindByKeyField method

Applies to TDBReference

Declaration

function FindByKeyField(V : Variant) : Boolean

Description

FindByKeyField method searches for the V parameter in the KeyField of the DataSet linking with the ListSource property (by calling the FindByKeyFieldInList method) or tries to execute the reference query at first if the UseQuery property is True. The method returns True if the record is found otherwise it returns False.

Example

```
if (DBReference1.FindByKeyField(varString (Edit1.Text))) then begin
  ...
end;
```

FindByKeyFieldInList method

Applies to TDBReference

Declaration

function FindByKeyFieldInList(V : Variant) : Boolean

Description

FindByKeyFieldInList method searches for the V parameter in the KeyField of the DataSet linking with the ListSource property. The method returns True if the record is found otherwise it returns False.

Example

```
if (DBReference1.FindByKeyFieldInList(varString (Edit1.Text))) then begin
   ...
end;
```

GotoKeyFieldValue method

Applies to TDBReference

Declaration

procedure GotoKeyFieldValue;

Description

Use the GotoKeyFieldValue method to synchronize the positions of the reference Query and the DataSet linking with the ListSourse property.

Example

```
if (DBReference1.State = rsValidated) then
   DBReference1.GotoKeyFieldValue;
```

RefreshText property

Applies to TDBReference

Declaration
procedure RefreshText;

Description
Update the reference text.

Example
DBReference1. RefreshText;

OnChange event

Applies to TDBReference

Declaration

property OnChange : TNotifyEvent;

Description

The OnChange event is activated when the text of the reference edit box or the Value of the AssignField of the TDBReference is changed.

Example

```
procedure TForm1.Reference1Change(Sender: TObject);
begin
  ShowMessage('The text   of the reference edit box is changed');
end;
```

TReference component

unit adbgrid

Description

The TReference component is a direct descendent of TWinControl.
The TReference is a very similar to the TDBReference but TReference doesn't have the DataField, DataSource and AssignField properties.
See TDBReference for more information.

TAutoDateEdit component

Properties

Unit aDatEdit

Description

The TAutoDateEdit component is a direct descendent of TEdit.
The TAutoDateEdit control has the bitbutton. On the onclick event of the bitbutton the date setting dialog form will be shown. The name of the months and days are fetched from the LOCALE_SMONTHNAME and LOCALE_SDAYNAME system locale entries.
The Date property is the date showing on the edit box.
If the IsDateNow property is True and the Date property is Nil then the current date is assigned to the Date property.
See also TAutoDBDateEdit.

Additional properties

property Date
property IsDateNow

TReferencePanel component (Client/Server suite only)

Properties

unit refer

Description

The TAutoDateEdit component is a direct descendent of TWinControl.
Only the reference controls use the ReferencePanel component. The ReferencePanel control displays on the bottom of the PopUp Form. You can place on it a Delphi or Filter controls.
To change the State property use the Component Editor.

Additional properties

property Style;

Style property

Applies to TReferencePanel

Declaration

property Style : TReferencePanelStyle;

Description

The State property specifies the style of the reference panel. There is the Component Editor for changing this property.
The possible values are those of the TReferencePanelStyle type:
1.bpComponent when the reference panel is shown on your like form the not control component.
2.bpStandart when the reference panel is shown on your form like all other controls.

Contacting Info

e-mail: rsd@tibc.tula.ru
INTERNET : HTTP:\\WWW.RSD.PP.RU
fax: +7 0872 770183

Authors:

1. Andrey Telnov and Roman Eremin - idea of creating the current project (Auto Components Library 32)
2. Andrey Telnov - programming
3. Roman Eremin and Dmitry Rogov - internal testing
4. Dmitry Rogov and Ura Sergeev - creating of the help system
5 Dmitry Rogov - creating Auto32 Expert
6. Ura Sergeev - technical support

Especial thanks for testing and suggestion to Theo Pistorius, M.D. Nazca Software (South Africa).