Auto Components Library (32-bit)

Version 1.15 (July 4, 1997)

Internet: http://www.rsd.pp.ru
E-mail: rsd@tibc.tula.ru
Fax: +7 (0872) 770183

We would like to present you some Delphi components. They are created after two year experiments with Delphi and their 16-bit version have been used in many applications. In this version some new components have been added and the others have been greatly changed. That is why we have decided not to release the previous (16-bit) version.
Now Auto Components Library distributes in two versions:

- Client/Server - 25 components + Auto Form Expert, US$120
- Desktop - 13 components, US$49

The components can be divided into four groups:
(The components marked with an asterisk are distributed with Client/Server version only)
.Data Access components:

- TAutoTable
- TAutoQuery*
- TAutoStoredProc*
- TAutoFind*
- TMacro (object)
- TMacros (object)

.Visual Filter Components Library:

- TAutoFilter (object)
- TCheckBoxFilter*
- TComboBoxFilter*
- TDateFilter
- TEasyFilter*
- TFilterLink
- TListBoxFilter*
- TLookupComboFilter
- TLookupListFilter*
- TMaskEditFilter*
- TRadioGroupFilter*
- TReferenceFilter*

Data Controls:

- TAutoDBGrid
- TAutoDBDateEdit
- TDBReference
- TReference
- TDBGridDateBtn
- TDBGridImageBtn

- TDBGridMemoBtn
- TDBGridRefBtn*

.Additional Components:

- TAutoDateEdit
- TReferencePanel*

The document consists of four sections:
I. Basic Features
II. Installation
III. Sales Notice
IV. The Development Team

I. Basic Features

1) Macros in Data Access Components
2) Visual Filter Components
3) Sorted TAutoDBGrid component
4) From TDBComboBox to TDBReference

1) Macros in Data Access Components

Traditionally in applying TTable and TQuery components we often have to change Filter and SQL properties at run-time which leads to error appearing and wasting time. To prevent these errors from appearing so often and to make the properties more convenient in applying we have created the TMacro object. The object is similar to TParam object but first of all the Filter property can also use the TMacro object and secondly the object has only two published properties: Name and Text. Before using the property where the TMacro object is declared it copies its Text property into the place where it is used.
For example: We have the following filter property - 'profit < &Summa', where 'summa' is the macro name which has the Text property assigned to '1000'. When the component begins to apply the Filter property its value will be changed to 'profit < 1000'.

2) Visual Filter Components

If you use the parameters of a dynamic SQL statement or the macros in applying AutoData Access Components you have to change the Value or the Text property at run-time. Usually changing the Values of the properties is the result of changing some property in Delphi controls. The user changes the properties of Delphi controls and your application has to refresh DataSet with the new values of TParam or TMacro properties. Using the Visual Filter Components you can create links between Macros or Params properties of the DataSet on one side and Controls on the other at design time.
The Visual Filter Components consists of TAutoFilter object, TFilterLink component and the set of components which have at least one property of the TAutoFilter type.
The TFilterLink component creates links between the AutoFilter object and the macro or the param object of the DataSet. On changing the Text property of the TAutoFilter, TFilterLink assigns its value to the value of macro or param objects and if DataSet is open and AutoRefresh property of TFilterLink is assigned to True (it is a default value) the TFilterLink reopens the cursor or refreshes the BDE filter of the DataSet.

For example: You need to create the possibility for users to fetch all the records from the table EMPLOYEE (Delphi's DBDEMOS) where the salary is greater than 'MinSalary' and the users may change the value 'MinSalary' in the application by typing in the necessary value.

1) We assign the filter property of DataSet to 'Salary>&MinSalary' or add to the SQL statement '...Salary>:MinSalary...' (or using macro '...Salary>&MinSalary...')
2) Drop TFilterLink and TMaskEditFilter to the form
3) Assign the TFilterLink properties:
    - DataSet to our DataSet
    - Filter to MaskEditFilter
    - Macro or Param to 'MinSalary'

It does not take a lot of time because to assign all these properties: you just need to select the necessary values from the drop-down lists. Now on changing text of the MaskEditFilter controls our Dataset will be refreshed.

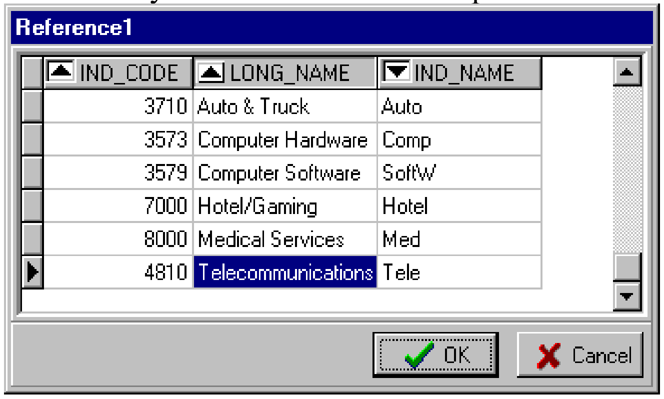
Notes:

1) If the filter component has more then one property of TAutoFilter type (for example, TDBReference has two such properties: AutoFilter and SQLOrderFilter) you have to assign the FilterName property to the necessary Value.
2) If the macro linked to the TFilterLink components is used only in the Filter property of DataSet then the TFilterLink refreshes only BDE filter without being reopen.

3) Sorted TAutoDBGrid component

TAutoDBGrid (see the picture) is inherited from TDBGrid and has two new features:

1. An ability to sort TDBGrid columns
2. An ability to perform incremental search on the current selected column by pressing a hot key.
3. An ability to change Color and Font of the rows.
4. An ability to view Memo and Graphic fields



IND_CODE	LONG_NAME	IND_NAME
3710	Auto & Truck	Auto
3573	Computer Hardware	Comp
3579	Computer Software	SoftW
7000	Hotel/Gaming	Hotel
8000	Medical Services	Med
4810	Telecommunications	Tele

1. The ability to sort TDBGrid columns

There are three types of column sorting order: Ascendant, Descendent, None (TSortedOrder = (gsNone, gsAsc, gsDesc)). There are three ways to make your column sorted:

a) add the '+' (for ascendant) or '-' (for descendent) character to the beginning of the TColumn DisplayText property
b) press Ctrl and Click the left mouse button at the column Title at Design time
c) call the procedure SetSortedOrder(ACol : Integer; ASortedOrder : TSortedOrder) at run-time.

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

a) If dataset is Table then TAutoDBGrid allows to sort columns if the linked fields are indexed. TAutoDBGrid sorts columns by changing the Value of the IndexName property.
b) If dataset is TAutoQuery then you have to create an ordering macro for the SQL property and link it to the Autofilter property of TAutoDBGrid using the filter link component (TFilterLink).

For example: the SQL statement is :

'Select name, full name, birth_date from Customer Order by &OrderField'. 'SQLOrder' is the macro name. Throw TFilterLink component to the form and assign its properties:

- DataSet to AutoQuery
- Filter to AutoDBGrid
- Macro to 'OrderField'

c) 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 cannot sort calculated fields.

2. The ability to perform incremental search in the currently selected column by pressing a hot key
To invoke the find dialog just press the hot key at run-time or call the 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 one by assigning the constant 'acDBGridHotKey' to your own hot key (unit aconst.pas).
3. The 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 than zero then the background color of the cell containing the 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. The 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.

4) 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 TDBReference's development history.
We worked under a project for one company in my city in Spring 1995. One of the main goals of the 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 a simple

document and more then half of the time was spent on searching the necessary record in TDBLookupCombo. At that time some DataSets had already had RecordCount equal to 100 - 300 records and it took from two to five seconds only to fetch them in spite 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 on the other hand the users sometimes knew the KeyValue and wanted to type it in instead of searching for it in the LookupCombo.
To reach the goal we decided to create our own component which would have the following features:

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

After such a component had been created we killed most of the 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 exceed 200 - 300 otherwise we apply TDBReference.

II. Installation

I. LIBRARY

A. Installation for Delphi 2.0

1.
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.

B. 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

C. Installation for C++ Builder 1.0

2. Copy the files from the \LIB.CB or (\LIB.D2 for registered users) to your C++ Builder Lib directory \LIB\DBTREE\
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. Delphi's VCL will now recompiles.

If on installation of the unregistered version the errors 'Could not open the file "*.obj"'
are appeared, then go to Options -> Environment -> Library and remove the check
'Options -> Use incremental linker' and rebuild library. Set the check after the library
rebuilding.

II. HELP FILES
1. Copy \help\autohelp.hlp, \help\ autohelp.kwf to \delhi\help\
2. Quit Delphi 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. If the error 'the autohelp.hlp file not found' appears point Delphi
to the delhi\help\autohelp.hlp.
The Auto Components Library help are now merged into Delphi's help system.

III Sales Notice
Auto Component Library (sources + support) costs:
- Client/Server - 25 components + Auto Form Expert, US$120
- Desktop - 13 components, US$49

We understand that our library is not a component set for interface improving and you have to test it before buying. So if you are not a registered user and want to test it you can do it without being afraid that the library stops working properly. Your program will work forever with our components. But we have added a protection to our library and the component (TAutoCheat, you can see it in our example programs) which removes the protection. It seems strange, but it is true. We hate cripple software. For more information about TAutoCheat see howcheat.txt.

Any comments and BUG reports are welcomed.
Copyright RSD software

E-mail: rsd@tibc.tula.ru
Internet: http://www.rsd.pp.ru
Fax: +7 (0872) 770183

IV The Development Team
Andrey Telnov and Roman Eremin - idea of creating the current project (Auto Component Library)

Andrey Telnov - programming
Roman Eremin and Dmitriy Rogov - internal testing
Dmitriy Rogov - the help system development
Dmitriy Rogov - the Auto Filter Expert development
Ura Sergeev - technical support

Thanks to all the people who have tested our library and sent us bug reports and comments.
Special thanks for testing and suggestions to Theo Pistorius, M.D. Nazca Software (South Africa).