

ImageLib 3.0

A fine product of SkyLine Tools

ImageLib DLL/VCL version 3.0 (c) Copyright 1995 by:

SkyLine Tools

Kevin Adams (CIS) 74742,1444

Jan Dekkers (CIS) 72130,353

Delphi Users' Guide

Technical support for C, C++, VB applications:

Kevin Adams: compuserve 74742,1444 or
Internet : 74742,1444@compuserve.com

Technical support for Delphi, Pascal and VB applications:

Jan Dekkers compuserve 72130,353 or
Internet : 72130,353@compuserve.com

Address:

SkyLine Tools

Attn: Jan Dekkers

11956 Riverside Drive 206

North Hollywood CA 91607

Phone 818 766-3900

Fax: 818 766-9027

What is ImageLib DLL/VCL?

The **ImageLib VCLs\DLL** is an inexpensive way to implement **JPEG, GIF, PCX, PNG, WMF, ICO, SCM.BMP, ICO and CMS** into your applications. **ImageLib** gives you a royalty free VCL with code included.

When ImageLib is being compiled into an executable application with the extension exe., then there are no licensing fees or royalties. Should any part of **ImageLib**, either the VCL or the DLL be used in a non-compiled application, such as: a value added VCL, VBX, OCX, royalties apply.

Other image and multimedia development tools are far more expensive than **ImageLib**. When users compared **ImageLib's** color resolution with other imaging tools, they found that **ImageLib's** was superior to most. In fact, the JPEG is so professional looking that museums are using it to catalogue their art.

Also, ImageLib adds DBMultiImage and DBMultiMedia to store and display **JPEG, BMP, GIF, PNG, CMS, SCM, PCX, AVI, MOV, MID, WAV and RMI** multimedia files in/from a TBlobField. International developers are able to display strings in the DLL as a resource file thereby enabling the translation into foreign languages.

SkyLine Tools stands behind its product with its highly responsive technical support.

ImageLib is an enhanced TImage and TDBImage VCL/DLL with the following added features:

- * Corrected Palette and Stretching of the Image Canvas (Delphi bug fix);
- * Enables the reading and writing of JPEG images to/from a file or a Tblobfield;
- * Enables the reading and writing of PNG images to/from a file or a Tblobfield;
- * Enables the reading and writing of PCX images to/from a file or a Tblobfield;
- * Enables the reading and writing of GIF images to/from a file or a Tblobfield;
- * Enables the reading and writing of BMP images to/from a file or a Tblobfield;
- * Jpeg 4, 8 and 24 bit dithering;
- * Jpeg 0 to 100% save quality;
- * Jpeg 0 to 100% smoothing;
- * PNG 4, 8 and 24 bit dithering;
- * GIF 4, 8 and 24 bit dithering;
- * PCX 4, 8 and 24 bit dithering;
- * BMP 4, 8 and 24 bit dithering;
- * BMP Reading from resources files;
- * Enables the reading and writing of Horizontal and Vertical Scrolling message images to/from a file or a Tblobfield;
- * Enables the reading and writing of AVI images to/from TBlobfield;
- * Enables the reading and writing of MOV images to/from TBlobfield;
- * Enables the reading and writing of WAV images to/from TBlobfield;
- * Enables the reading and writing of RMI images to/from TBlobfield;
- * Enables the reading and writing of MID images to/from TBlobfield;
- * Enables the reading and writing of ICO images to/from a file(Delphi inherited);
- * Enables the reading and writing of WMF images to/from a file (Delphi inherited);
- * TMultimage CUT/COPY and Paste to/from the clipboard;
- * All Multi VCL's have full Print Support with 1 line of code;

- * Internal scrolling message editor;
- * DLL Callback function, to show a progress bar and to process Messages;
- * No code necessary (VCL) to display all image formats from a TBlobfield;
- * Loads/Saves all Tblobfield images to/from file;
- * Converts all Tblobfield images to Jpeg/Bmp/Gif/Pcx/Png file;
- * Pastes images from Clipboard and stores as a Jpeg/Bmp/Gif/Pcx/Png file/Blob;
- * Foreign error strings. DLL strings are stored in the DLL resource
- * Full VCL source code provided without extra charge

Installation Instructions

BACKUP YOUR \DELPHI\BIN\COMPLIB.DCL Better safe than sorry.

Copy the IMGLIB30.DLL to a directory on your path or to the windows\system directory. IMGLIB30.DLL is a DISTRIBUTABLE FILE and needs to be included with your application.

Unzip the EXAMPLS.ZIP into a new directory. Copy the following files into a directory containing your 3rd party added VCL's: (If you don't have a directory yet please, make one).

WHEN EVALUATING

PMREG.PAS, PMREG.DCR, TMULTIP.PAS, TDMULTIP.PAS, DLL30.DCU, SETSR30.DFM, SETSR30.PAS, SETCR30.DFM and SETCR30.PAS

WHEN PURCHASED

PMREG.PAS, PMREG.DCR, TMULTIP.PAS, TDMULTIP.PAS, DLL30.PAS, SETSR30.DFM, SETSR30.PAS, SETCR30.DFM and SETCR30.PAS

Copy IMGLIB30.HLP to your \DELPHI\BIN directory.
Copy IMGLIB30.KWF to your \DELPHI\HELP directory.

Execute delphi's HelpInst and open delphi.hdx which is located in your \DELPHI\BIN. Add a new Keyword file by the name IMGLIB30.KWF. Save delphi.hdx which will automatically rebuild your delphi help file.

Execute Delphi. In Delphi select Options\Install components\ Add and browse your 3rd party added VCLs directory. Select PMREG.PAS and press the OK button.

After the library is rebuilt, you will notice 4 new icons on your Delphi toolbar under images called:

Multimage.

DBMultimage,
DBMultiMedia,
DBMediaPlayer.

Troubleshooting:

The Delphi Library searchpath is very short (127 characters). The more VCL components you add, the larger your searchpath. Should you get a message PMREG.PAS or PMREG.DCU not found, then your path is being truncated, the solution is to copy several 3rd party VCLs into one directory and delete the freed directories from your searchpath. If Complib cannot find IMGLIB30.DLL you will notice that all Icons are gone from your delphi toolbar and you get a message COMPLIB.DCL not found. No Panic, Just copy IMGLIB30.DLL to a directory on your path or to the windows\system directory and restore your backed up complib.

Installation Instructions for the Examples

In delphi select Open\Project and open one of the projects in the newly created directory. Select rebuild. Run the program.

IF YOU INSTALLED THE OLD MULTIIMAGE or DBMULTIIMAGE

What to do with your existing programs using the old Multimage VCL:

In case of OLD MULTIIMAGE:

Change the uses clause of your programs from REG_IMAG or REG_IM20 or TMULTI to TMULTIP, which is the replacement for REG_IMAG or REG_IM20 or TMULTI.

In case of OLD TDBMULTIIMAGE:

Change the uses clause of your programs from REG_IM20 or TDBMULTI to TDMULTIP which is the replacement for REG_IMAG or REG_IM20 or TDBMULTI.

(Only for update from version 1.0 to version 2.0)

When you startup your existing programs using the Multimage VCL you might notice a complain (Property JPegSaveSmooH doesn't exist or Property JPegSaveFileName doesn't exist).

Property JPegSaveSmooH is renamed to JPegSaveSmooTh (watch the T).
To fix this, Load the FORM (the *.DFM) file complaining about this and replace JPegSaveSmooH with JPegSaveSmooTh (add the T).

Property JPegSaveFileName is renamed to DefSaveFileName.

To fix this, Load the FORM (the *.DFM) file complaining about this and replace JPegSaveFileName with DefSaveFileName

New added Visual Components

The new VCL objects added to your toolbar are called:

Multimage,
DBMultimage,
DBMultiMedia
DBMediaPlayer.

TMULTIIMAGE: JPEG, BMP, GIF, PNG, WMF, SCM, CMS, ICO and PCX.

Sample projects:

im_cvrt.dpr Converting images example
scrollim.dpr Scrolling messages example
simple.dpr A few lines of code example
viewph.dpr Extensive example

Multimage has the same properties as Delphi's TImage with the following additions:
Reading and displaying images for all Image formats

property ImageName

Visual property

Value

Filename of the image which needs to be displayed.

Purpose

JPEG, BMP, GIF, PNG, WMF, SCM, CMS, ICO and PCX. images are loaded with one single line of code.

Example

Multimage1.Imagename:=C:\ CLOWN.JPG';

JPEG File read and write

property JPegSaveQuality _____

Visual property

Value

0...100

Purpose

0 is poor and 100 excellent. We normally use 25 to have a reasonable quality with 1/10 savings in size.

Example

MultImage1.JPEGSaveQuality:=25;

property JPEGSaveSmooth

Visual property

Value 0...100

Purpose

0 is no smoothing and 100 is full smoothing. Because of the lossy compression of Jpegs, an image might be too hard; smoothing can give it a better look.

Example

MultImage1.JPEGSaveSmooth:=5;

procedure SaveAsJpg(FN : TFilename);

Value

Filename of the file saved to

Purpose

Save the displayed image to a jpeg file.

Remark

An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName

Example

```
procedure TForm1.SaveButtonClick(Sender: TObject);  
begin
```

```
if SaveDialog1.execute then begin
    MultImage1.JPEGSaveSmooth:=5;
    MultImage1.JPEGSaveQuality:=25;
    MultImage1.SaveAsJpg(SaveDialog1.FileName);
end;
end;
```

property DefSaveFileName

visual property

(Changed from JPJSaveFileName in version 2.0)

Value

Filename of the BMP, JPG, GIF, PCX, PNG which needs to be saved.

Purpose

It can come in handy to store a filename long before the file is actually saved. You can use this as a filename scratchpad.

Example

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
if SaveDialog1.execute then begin
    MultImage1.JPEGSaveQuality:=25;
    MultImage1.JPEGSaveSmooth:=5;
    MultImage1.DefSaveFileName:=SaveDialog1.FileName;
    MultImage1.SaveAsJpg("");
end;
end;
```

property ImageDither

Visual property

Value

True or False

Purpose

Dithering is used in conjunction with the **ImageReadRes**. If ImageReadRes = ColorTrue then **ImageDither** is not used
ImageReadRes = Color256 then **ImageDither** False or True for dither option
ImageReadRes = Color16 then **ImageDither** False or True for dither option

In all cases dithering is only used if it has to change resolutions of the input image. If a resolution of **Color256** is specified and the input image is already **256 colors** then the dithering will do nothing. If the input is **ColorTrue** and VGA resolution is **256 colors** then the image will be dithered if set to true.

Example

```
procedure TForm1.DoImage;  
begin  
    MultiImage1._ImageDither:=True;  
    MultiImage1.ImageReadRes:= Color256;  
    MultiImage1.imageName:='c:\frog.jpg';  
end;  
end;
```

property ImageReadRes

Visual property

Value

ColorTrue, Color256 or Color16

Purpose

To force an image to be read in a specific resolution. Lets assume that the VGA display of a particular computer is 16 colors but the Image is a 256 color image. This image needs to be color reduced to be shown on the 16 color PC.

Example

```
procedure TForm1.OpenFileClick(Sender: TObject);  
begin  
    if OpenFileDialog1.execute then begin  
        MultiImage1._ImageDither:=True;  
        MultiImage1.ImageReadRes:= Color256;  
        MultiImage1.imageName:=OpenDialog1.filename;  
    end;  
end;
```

property ImageWriteRes

Visual property

Value

ColorTrue, Color256 or Color16

Purpose

To force an image to be written in a specific resolution (Upscale or Downscale)

Example

```
procedure TForm1.SaveFileClick(Sender: TObject);
begin
  if SaveDialog1.execute then begin
    MultImage1. ImageWriteRes:= Color16;
    MultImage1.SaveAsBMP(SaveDialog1.FileName);
  end;
end;
```

BMP File read and write

To read/display a BMP image you can use either Imagelib or Delphi

Example using the Delphi way.

This example uses two picture components. When the form first appears, two bitmaps are loaded into the picture components and stretched to fit the size of the components. To try this code, substitute names of bitmaps you have available.

The following code will load BMP, WMF and ICO Images

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  MultImage1.Stretch := True;
  MultImage2.Stretch := True;
  MultImage 1.Picture.LoadFromFile('BITMAP1.BMP');
  MultImage 2.Picture.LoadFromFile('BITMAP2.BMP');
end;
```

Example using the ImageLib way.

This example uses two picture components. When the form first appears, two bitmaps are loaded into the picture components and stretched to fit the size of the components. To try this code, substitute names of bitmaps you have available.

The following code will load JPEG, BMP, SCM, GIF, WMF, ICO and PCX Images

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  MultiImage1.Stretch := True;
  MultiImage2.Stretch := True;
  MultiImage 1.ImageName:='BITMAP1.BMP';
  MultiImage 2.ImageName:='BITMAP2.BMP';
end;
```

To Save a BMP image you can use either Imagelib or Delphi

Example using the Delphi way.

This example uses two picture components.

```
begin
  MultiImage1.Picture.SaveToFile('BITMAP1.BMP');
  MultiImage2.Picture.SaveToFile('BITMAP2.BMP');
end;
```

Saving BMP's the ImageLib way.

procedure SaveAsBMP(FN : TFilename);

Value

Filename of the file to which it is being saved.

Purpose

Save the displayed image to a bmp file.

Remark

An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName

Example

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
if SaveDialog1.execute then begin
    MultiImage1.DefSaveFileName:=SaveDialog1.FileName;
    MultiImage1.SaveAsBMP("");
end;
end;
```

Or

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
if SaveDialog1.execute then
    MultiImage1.SaveAsBMP(SaveDialog1.FileName);
end;
```

GIF File read and write

Gif uses LZW compression which is patented by Unisys. On CompuServe GO PICS to obtain information about the Unisys patents. In order to use the ImageLib's GIF read and write, you need to buy a license from Unisys. By using ImageLib's GIF Read and Write features you acknowledge that SkyLine has notified you about the LZW patent and hold SkyLine harmless from any legal actions.

property ImageName

Visual property

Value

Filename of the image which needs to be displayed.

Purpose

JPEG, BMP, GIF, PNG, WMF, SCM, CMS, ICO and PCX images are loaded with one single line of code.

Example

```
MultiImage1.Imagename:=C:\ CLOWN.GIF;
```

procedure SaveAsGIF(FN : TFilename);

Value

Filename of the file to which it is being saved.

Purpose

Save the displayed image to a GIF file.

Remark

An active image need to be displayed on the form. If no filename is passed it will use the DefSaveFileName

Example

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
  if SaveDialog1.execute then
    MultiImage1.SaveAsGIF(SaveDialog1.FileName);
end;
```

PCX File read and write**property ImageName**

Visual property

Value

Filename of the image which needs to be displayed.

Purpose

JPEG, BMP, GIF, PNG, WMF, SCM, CMS, ICO and PCX images are loaded with one single line of code.

Example

```
MultiImage1.Imagename:=C:\ CLOWN.PCX;
```

procedure SaveAsPCX(FN : TFilename);**Value**

Filename of the file to which it is being saved.

Purpose

Save the displayed image to a PCX file.

Remark

An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName

Example

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
  if SaveDialog1.execute then
    MultiImage1.SaveAsPCX(SaveDialog1.FileName);
end;
```

PNG File read and write

property ImageName

Visual property

Value

Filename of the image which needs to be displayed.

Purpose

JPEG, BMP, GIF, PNG, WMF, SCM, CMS, ICO and PCX. Images are loaded with one single line of code.

Example

```
MultiImage1.Imagename:=C:\ CLOWN.PNG;
```

procedure SaveAsPNG(FN : TFilename);

Value

Filename of the file to which it is being saved.

Purpose

Save the displayed image to a PNG file.

Remark

An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName

Example

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
  if SaveDialog1.execute then
    MultImage1.SaveAsPNG(SaveDialog1.FileName);
end;
```

Credit Messages File read and write

Overview

Credit messages are TMultimages created by the VCL on the fly. The average filesize of a Credit message (CMS) is only 200 bytes. The maximum size is 64Kb. Stored in the CMS file are:

MessageFont	: TFont;	the message's font
MessageSpeed	: Integer;	the scrolling speed 1 is fast 10 is slow
MessageColor	: TColor;	the background color
CreditBoxList	: TStringList;	the credit messages in a stringlist

The VCL does NOT have its own moving engine. You "the programmer" must trigger the movements. The reason for this is that an application can have only one Application.OnIdle event. This event then needs to be shared by other events which may need a trigger. Note that other VCLs could also use a Trigger. Make sure that their OnIdle proc. doesn't destroy MultImage's trigger.

In your application you need to add a procedure to the private clauses called, for instance, Trigger:

```
type
  TForm1 = class(TForm)
  procedure FormCreate(Sender: TObject);
  private
    Procedure Trigger(Sender : TObject; Var Done : Boolean);
  public
```

end;

In the form create you will assign Trigger to the onIdle event.

```
procedure Form1.FormCreate(Sender: TObject);
begin
    Application.OnIdle:=Trigger;
end;
```

The procedure trigger will then trigger the VCL:

```
Procedure Form1.Trigger(Sender : TObject; Var Done : Boolean);
begin
    MultImage3.Trigger;
    MultImage2.Trigger;
    MultImage1.Trigger;
end;
```

For an extensive example load the project Scrollim.dpr

Procedure Trigger;

Value

None

Purpose

Trigger the scrolling message movements.

Example

```
Procedure TForm1.Trigger(Sender : TObject; Var Done : Boolean);
begin
    MultImage1.Trigger;
end;
```

procedure CreateCreditMessage(MessagePath : String; AutoLoad : boolean);

Value

MessagePath

The initial path displayed in the save dialog.

AutoLoad

True or False. If true, message is displayed after saving it.

Purpose

CreateCreditMessage will open the Message editor. The user can create his own Credit message and save this message to a file with a CMS extension as default.

Example

```
procedure TForm1.BitBtn2Click(Sender: TObject);
```

```
begin
  MultImage1.CreateCreditMessage(ExtractFilePath(Application.Exename), True);
end;
```

procedure SaveCurrentCreditMessage(MessageName : TFileName);

Value

MessageName The filename to which the message is being saved.

Purpose

Save the message with values of: (These are the values of the current message being displayed).

MultImage1.CreditBoxList	: TStringList;	The credit messages in a stringlist
MultImage1.MessageFont	: Tfont;	The message font
MultImage1.MessageColor	: Tcolor;	Background color
MultImage1.MessageSpeed	: Integer;	Scrolling Speed

Example

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  MultImage1.FreeMsg;
  MultImage1.CreditBoxList.Clear;
  MultImage1.CreditBoxList.Add(' ImageLib');
  MultImage1.CreditBoxList.Add(' Another fine product of');
  MultImage1.CreditBoxList.Add(' SKYLINE TOOLS');
  MultImage1.CreditBoxList.Add(' Programming : Kevin Adams');
  MultImage1.CreditBoxList.Add(' Programming : Jan Dekkers');
  MultImage1.CreditBoxList.Add(' Artwork & PR: Jillian Pinsker');
  MultImage1.MessageFont.Name:='Arial';
  MultImage1.MessageFont.Size:=-40;
  MultImage1.MessageFont.Style:=[fsitalic, fsbold];
```



```
MultImage1.MessageFont.Color:=clWhite;
MultImage1.MessageColor:=clNavy;
MultImage1.MessageSpeed:=1;
if SaveDialog1.Execute then
  MultImage1.SaveCurrentCreditMessage(SaveDialog1.FileName);
end;
```

Remark

MessageFont.Name, MessageFont.Size, MessageFont.Style and MessageFont.Color could also be defined using a fontdialog box :

Example

```
MultImage1.MessageFont:= FontDialog1.Font;
```

procedure NewCreditMessage;

Value

None

Purpose

Initiate a new message. Ideal to show messages created on the fly.

Example

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  MultImage1.FreeMsg;
  MultImage1.CreditBoxList.Clear;
  MultImage1.CreditBoxList.Add(' ImageLib');
  MultImage1.CreditBoxList.Add(' Another fine product of');
  MultImage1.CreditBoxList.Add(' SKYLINE TOOLS');
  MultImage1.CreditBoxList.Add(' Programming : Kevin Adams');
  MultImage1.CreditBoxList.Add(' Programming : Jan Dekkers');
  MultImage1.CreditBoxList.Add(' Artwork & PR: Jillian Pinsker');
  MultImage1.MessageFont.Name:='Arial';
  MultImage1.MessageFont.Size:=-40;
  MultImage1.MessageFont.Style:=[fsitalic, fsbold];
  MultImage1.MessageFont.Color:=clWhite;
  MultImage1.MessageColor:=clNavy;
  MultImage1.MessageSpeed:=1;
  MultImage1.NewCreditMessage;
end;
```

Procedure FreeMsg;

Value

None

Purpose

Disposes the current message and assigns the Picture to Nil

Example

```
procedure TForm1.BitBtn5Click(Sender: TObject);
begin
    Multimage1.FreeMsg;
end;
```

Scrolling Messages File read and write

Overview

Scrolling messages are TMultiimages created by the VCL on the fly. The average file size of a Scrolling message (SCM) is only 200 bytes. Stored in the SCM file are:

MessageText	: String;	The message text.
MessageFont	: Tfont;	The message font.
MessageColor	: Tcolor;	Background color.
MessageSpeed	: Integer;	Scrolling Speed.

The VCL does NOT have its own moving engine. You "the programmer" must trigger the movements. The reason for this is that an application can have only one Application.OnIdle event. This event then needs to be subdivided to other events which may need an Idle event. Note that other VCLs could also use a Trigger. Make sure that their OnIdle proc. doesn't destroy Multimage's trigger.

In your application you need to add a procedure to the private clauses called, for instance, Trigger:

```
type
    TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
    private
        Procedure Trigger(Sender : TObject; Var Done : Boolean);
    public
```

end;

In the form create you will assign Trigger to the onIdle event.

```
procedure Form1.FormCreate(Sender: TObject);
begin
    Application.OnIdle:=Trigger;
end;
```

The procedure trigger will then trigger the VCL:

```
Procedure Form1.Trigger(Sender : TObject; Var Done : Boolean);
begin
    MultImage3.Trigger;
    MultImage2.Trigger;
    MultImage1.Trigger;
end;
```

For an extensive example load the project Scrollim.dpr.

Procedure Trigger;

Value

None

Purpose

Trigger the scrolling message movements.

Example

```
Procedure TForm1.Trigger(Sender : TObject; Var Done : Boolean);
begin
    MultImage1.Trigger;
end;
```

procedure CreateMessage(MessagePath : String; AutoLoad : Boolean);

Value

MessagePath

The initial path displayed in the save dialog.

AutoLoad

True or False. If true, message is displayed after saving it.

Purpose

CreateMessage will open the Message editor. The user can create his own scrolling message and save this message to a file with an SCM extension as default.

Example

```
procedure TForm1.BitBtn2Click(Sender: TObject);
```

```
begin
    MultImage1.CreateMessage(ExtractFilePath(Application.Exename), True);
end;
```

procedure SaveCurrentMessage(MessageName : TFileName);

Value

MessageName The filename to which the message is being saved.

Purpose

Save the message with values of: (These are the values of the current message being displayed).

MultImage1.MessageText	: String;	The message text.
MultImage1.MessageFont	: Tfont;	The message font
MultImage1.MessageColor	: Tcolor;	Background color
MultImage1.MessageSpeed	: Integer;	Scrolling Speed

Example

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
    MultImage1.MessageText:='ImageLib A great tool ';
    MultImage1.MessageFont.Name:='Arial';
    MultImage1.MessageFont.Size:=-40;
    MultImage1.MessageFont.Style:=[fsitalic, fsbold];
    MultImage1.MessageFont.Color:=clWhite;
    MultImage1.MessageColor:=clNavy;
    MultImage1.MessageSpeed:=1;
    if SaveDialog1.Execute then
        MultImage1.SaveCurrentMessage(SaveDialog1.FileName);
end;
```

Remark

MessageFont.Name, MessageFont.Size, MessageFont.Style and MessageFont.Color could also be defined using a FontDialog box.

Example

MultImage1.MessageFont:= FontDialog1.Font;

procedure NewMessage;**Value**

None

Purpose

Initiate a new message. This is ideal to show messages created on the fly.

Example

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
    MultImage1.MessageText:='ImageLib 3.0 A great tool ';
    MultImage1.MessageFont.Name:='Arial';
    MultImage1.MessageFont.Size:=-40;
    MultImage1.MessageFont.Style:=[fsitalic, fsbold];
    MultImage1.MessageFont.Color:=clWhite;
    MultImage1.MessageColor:=clNavy;
    MultImage1.MessageSpeed:=1;
    MultImage1.NewMessage;
end;
```

Procedure FreeMsg;**Value**

None

Purpose

Dispose the current message and then assign Picture to Nil

Example

```
procedure TForm1.BitBtn5Click(Sender: TObject);  
begin  
    MultImage1.FreeMsg;  
end;
```

CLIPBOARD

procedure CopyToClipboard;

Value

None

Purpose

Copy the current displayed image to the clipboard

Example

```
procedure TForm1.Copy1Click(Sender: TObject);  
begin  
    MultImage1.CopyToClipboard;  
end;
```

procedure CutToClipboard;

Value

None

Purpose

Copy the current displayed image to the clipboard and erase it from the canvas.

Example

```
procedure TForm1.Cut1Click(Sender: TObject);  
begin  
    MultImage1.CutToClipboard  
end;
```

procedure PasteFromClipboard;

Value

None

Purpose

Paste an image from the clipboard into the Multimage.

Example

```
procedure TForm1.Paste1Click(Sender: TObject);  
begin  
    MultImage1.PasteFromClipboard;
```

end;

Printing Multimage Images

Tmultimage has full printing support to print JPEG, GIF, BMP, PCX, WMF and ICO. It does this with one procedure call.

procedure PrintMultimage(X, Y, pWidth, pHeight: Integer);

Value

X	The Left position of the image on the paper
Y	The Top position of the image on the paper
pWidth	The Right position of the image on the paper
pHeight	The Bottom position of the image on the paper

Purpose

PrintMultimage will Stretch the image on the Printer.Canvas and print it.

Remark

Icons can't be stretched and will be printed in their original size.

If pWidth and/or pHeight are 0, then the image will be printed in its original size.

Example

```
procedure TForm1.Print1Click(Sender: TObject);
begin
  if PrintDialog1.execute then
    Multimage1.PrintMultimage(0, 0, 0, 0);
end;
```

Image Information

Function GetInfoAndType(filename : TFilename) : Boolean;

Value

Filename of the image

Purpose

GetInfoAndType is a very fast function which retrieves image information without actually loading the complete image.

Returns

True, if successful, otherwise False. GetInfoAndType will store the following information:

For all filetypes:

Bfiletype : String; Return: JPEG, BMP, GIF, PNG, PCX, ICO, WMF, SCM, CMS
Bwidth : Integer; Return: Width of the image
BHeight : Integer; Return: Height of the image
BSize : Longint; Return: File size in bytes
Bcompression : String; Return: Compression method

For JPEG, BMP, GIF, PCX, PNG only (ICO, WMF, SCM, CMS will return 0)

Bbitspixel : Integer; Return: Bits per Pixel
Bplanes : Integer; Return: Planes
Bnumcolors : Integer; Return: Number of colors

Remark

GetInfoAndType is called automatically by the VCL during an Image load. If no Image is displayed you can call this function manually.

Example

```
procedure TForm1.DisplayInfo(filename : TFilename);
begin
  if GetInfoAndType(filename) then begin
    Edit1.Text:=IntToStr(MultImage1.Bwidth);
    Edit2.Text:=IntToStr(MultImage1.BHeight);
    Edit3.Text:=IntToStr(MultImage1.Bbitspixel);
    Edit4.Text:=IntToStr(MultImage1.Bplanes);
    Edit5.Text:=IntToStr(MultImage1.Bnumcolors);
    Edit6.Text:=MultImage1.BFileType;
    Edit7.Text:=MultImage1.Bcompression;
    Edit8.Text:=IntToStr(MultImage1.BSize)+ ' bytes';
  end;
end;
```

DLL Image CallBack Procedure

(Changed in version 2.2 from a procedure to a function).

(Changed in version 2.2.1 from to use a C calling convention).

Overview

The callback procedure is generated by the DLL and has 3 main goals:

- 1: To show a progress bar to the user
- 2: To process windows messages to give other windows programs the chance to do what they have to do.
- 3: To inform the DLL that either everything is OK or to cancel the operation

It's up to you, the application developer, to process the application's message loop. You can do this by adding `APPLICATION.PROCESSMESSAGES` in the callback procedure.

The DLL expects the following type of callback function to be registered:

TCallbackFunction = function (I : Integer) : cdecl Integer;

Value

You need to pass a 1 if O.K. or a 0 if you want to cancel

Returns

A value between 1 and 100 which identifies the progress of the image being loaded.

Remarks and Example

There are two things you *MUST* do to add a callback to your app:

- 1: You need to declare a function of the type above with the `EXPORT` and `cdecl` clause:

```
Function ImageLibCallBack(i : integer) : integer; cdecl; export;
begin
  if Application.Terminated then
    Result:=0
  else begin
    Application.ProcessMessages;
    Form1.Gauge1.Progress:=i;
    Result:=1;
  end;
end;
```

2: You need to register the callback to the VCL. The best place to do that is in the FormCreate function:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TMultImageCallBack:= ImageLibCallBack;
end;
```

TDBMULTIIMAGE: Sample project Blob.dpr

Displays and stores JPEG, BMP, GIF, PNG, SCM, CMS and PCX from/to a TBLOBField.

TDBMultimage is the data-aware VCL version of TMultImage. DBMultimage is derived from TCustomControl. It has the same properties as Delphi's TDBImage with the following additions:

property JPegSaveQuality _____

Visual Property

Value

0...100

Purpose

0 is poor and 100 excellent. We normally use 25 to have a reasonable quality with a 1/10 saving in size.

Example

DBMultImage1.JPegSaveQuality:=25;

property JPegSaveSmooth _____

Visual Property

Value

0...100

Purpose

0 is no smoothing and 100 is full smoothing. Because of the lossy compression of Jpegs, an image might be too hard, smoothing can give it a better look.

Example

```
DBMultiImage1.JPegSaveSmooth:=5;
```

procedure SaveToFileAsJpg(FN : TFilename);

Value

The filename of the Jpeg to which the image is being saved .

Purpose

To saves the image displayed as a Jpeg file.

Remark

Image must be displayed

Example

```
procedure TForm1.BitBtn8Click(Sender: TObject);
```

```
begin
```

```
    DBMultiImage1.JPegSaveQuality:=25;
```

```
    DBMultiImage1.JPegSaveSmooth:=5;
```

```
    If SaveDialog2.Execute then
```

```
        DBMultiImage1.SaveToFileAsJpeg(SaveDialog2.Filename);
```

```
end;
```

property ImageDither

Visual property

Value

True or False

Purpose

Dithering is used in conjunction with the **ImageReadRes**. If
ImageReadRes = ColorTrue then **ImageDither** is not used
ImageReadRes = Color256 then **ImageDither** False or True for dither option
ImageReadRes = Color16 then **ImageDither** False or True for dither option

In all cases dithering is only used if it has to change resolutions of the input image. If a resolution of **Color256** is specified and the input image already has **256 colors** then the dithering will do nothing. If the input is **ColorTrue** and VGA resolution is **256 colors** then the image will be dithered if set to true.

Example

```
DBMultImage1.ImageDither:=True;  
DBMultImage1.ImageReadRes:= Color256;
```

property ImageReadRes

Visual property

Value

ColorTrue, Color256 or Color16

Purpose

To force an image to be read at a specific resolution. Let's assume that the VGA display of a particular computer is 16 colors but the Image is a 256 color image. This image needs to be color reduced to be shown on the 16 color PC.

Example

```
DBMultImage1.ImageDither:=True;  
DBMultImage1.ImageReadRes:= Color256;
```

property ImageWriteRes

Visual property

Value ColorTrue, Color256 or Color16

Purpose

To force an image to be written in a specific resolution (Upscale or Downscale)

Example

```
DBMultImage1.ImageWriteRes:= Color16;
```

procedure SaveToFileAsBMP(FN : TFilename);

Value

The filename of the BMP to which the image is being saved.

Purpose

To save the Image displayed as a BMP file.

Remark

Image must be displayed

Example

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
    DBMultiImage1.ImageWriteRes:= Color256;
    If SaveDialog2.Execute then
        DBMultiImage1. SaveToFileAsBMP(SaveDialog2.Filename);
end;
```

procedure SaveToFileAsGIF(FN : TFilename);

Value

The filename of the GIF to which the image is being saved.

Purpose

To saves the Image displayed as a GIF file.

Remark

Image must be displayed

Example

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
    DBMultiImage1.ImageWriteRes:= Color16;
    If SaveDialog2.Execute then
        DBMultiImage1. SaveToFileAsGIF(SaveDialog2.Filename);
end;
```

procedure SaveToFileAsPCX(FN : TFilename);

Value

The filename of the PCX to which the image is being saved.

Purpose

To save the Image displayed as a PCX file.

Remark

Image must be displayed

Example

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
    DBMultImage1.ImageWriteRes:= ColorTrue;
    If SaveDialog2.Execute then
        DBMultImage1. SaveToFileAsPCX(SaveDialog2.FileName);
end;
```

procedure SaveToFileAsPNG(FN : TFilename);

Value The filename of the PNG to which the image is being saved.

Purpose

To save the Image displayed as a PNG file.

Remark

Image must be displayed

Example

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
    DBMultImage1.ImageWriteRes:= Color256;
    If SaveDialog2.Execute then
        DBMultImage1. SaveToFileAsPNG(SaveDialog2.FileName);
end;
```

procedure SaveToFile(filename : TFilename);**Value**

The filename of the file to which it is being saved.

Purpose

Saves the current blob to a file AS Stored (**No conversion**)

Example

```
procedure TForm1.BitBtn2Click(Sender: TObject);
var temp : string;
begin
    temp:=DBMultImage1.GetInfoAndType;
    if temp = 'GIF' then begin
        SaveDialog1.filter:='GIF files|*.GIF';
        SaveDialog1.DefaultExt:='GIF';
    end else if temp = 'PCX' then begin
```

```

        SaveDialog1.filter:='PCX files|*.PCX';
        SaveDialog1.DefaultExt:='PCX';
end else if temp = 'PNG' then begin
    SaveDialog1.filter:='PNG files|*.PNG';
    SaveDialog1.DefaultExt:='PNG';
end else if temp = 'JPG' then begin
    SaveDialog1.filter:='Jpeg files|*.JPG';
    SaveDialog1.DefaultExt:='JPG';
end else if temp = 'BMP' then begin
    SaveDialog1.filter:='BMP files|*.BMP';
    SaveDialog1.DefaultExt:='BMP';
end else if temp = 'SCM' then begin
    SaveDialog1.filter:='SCM files|*.SCM';
    SaveDialog1.DefaultExt:='SCM';
end;

If SaveDialog1.Execute Then
    DBMultImage1.SaveToFile(SaveDialog1.FileName);
end;

```

Image Information

Function GetInfoAndType : String

Value

None

Purpose

GetInfoAndType is a very fast function which retrieves image information without actually loading the complete image.

Returns

Extension format of the file stored in the blobfield. GetInfoAndType will store the following information:

For all filetypes:

Bfiletype	: String;	Return: JPEG, BMP, GIF, PCX, ICO, WMF, SCM,CMS,PNG
Bwidth	: Integer;	Return: Width of the image
BHeight	: Integer;	Return: Height of the image
BSize	: Longint	Return: File size in bytes
Bcompression	: String;	Return: Compression method

For JPEG, BMP, GIF, PCX, PNG only (ICO, WMF, SCM, CMS will return 0)

Bbitapixel	: Integer;	Return: Bits per Pixel
------------	------------	------------------------

Bplanes : Integer; Return: Planes
Bnumcolors : Integer; Return: Number of colors

Remark

GetInfoAndType is called automatically by the VCL during an Image load (if autodisplay is true). If no Image is displayed or autodisplay is false you can call this function manually.

Example

```
procedure TForm1.DataSource1DataChange(Sender: TObject; Field: TField);
begin
  If not DBMultilImage1.autodisplay then DBMultilImage1.GetInfoAndType;
  Edit1.text:='This blob image is a '+TDBMultilImage1.BFiletype;
  Edit2.text:=IntToStr(DBMultilImage1.Bwidth);
  Edit3.text:=IntToStr(DBMultilImage1.BHeight);
  Edit4.text:=IntToStr(DBMultilImage1.Bbitspixel);
  Edit5.text:=IntToStr(DBMultilImage1.Bplanes);
  Edit6.text:=IntToStr(DBMultilImage1.Bnumcolors);
  Edit7.text:=TDBMultilImage1.Bcompression;
  Edit8.text:=IntToStr(DBMultilImage1.BSize);
end;
```

property UpdateAsJPG : Boolean

property UpdateAsBMP : Boolean

property UpdateAsGIF : Boolean

property UpdateAsPCX : Boolean

property UpdateAsPNG : Boolean

Visual properties

Value

True or False

Purpose

To store a new image or to update the displayed image. If True then the Blob Image will be updated to a Blob in one of the formats above which is set to true.

Remark

Image must be displayed

Example

```
procedure TForm1.UpdateAsJpeg(Sender: TObject);
begin
  DBMultilImage1.UpdateAsJpeg:=True;
  DBMultilImage1.PastefromClipboard;
  Table1.Post;
end;
```


Printing DBMultimage Images

TDBmultimage has full printing support to print JPEG, GIF, BMP, PCX, PNG. It does this with one procedure call.

procedure PrintMultimage(X, Y, pWidth, pHeight: Integer);

Value

X	The Left position of the image on the paper
Y	The Top position of the image on the paper
pWidth	The Right position of the image on the paper
pHeight	The Bottom position of the image on the paper

Purpose

PrintMultimage will Stretch the image on the Printer.Canvas and print it.

Remark

Icons can't be stretched and will be printed in their original size.
If pWidth and/or pHeight are 0, the image will be printed in its original size.

Example

```
procedure TForm1.Print1Click(Sender: TObject);
begin
  if PrintDialog1.execute then
    DBMultimage1.PrintMultimage(0, 0, 0, 0);
end;
```

CLIPBOARD

procedure CopyToClipboard;

Value

None

Purpose

Copy the currently displayed image to the clipboard

Remark

CRTL INSERT and CRTL C does the same

Example

```
procedure TForm1.Copy1Click(Sender: TObject);
begin
  DBMultimage1.CopyToClipboard;
end;
```

procedure CutToClipboard;

Value

None

Purpose

Copy the currently displayed image to the clipboard and erase it.

Remark

SHIFT DELETE and CTRL X does the same.

Example

```
procedure TForm1.Cut1Click(Sender: TObject);
begin
    DBMultiImage1.CutToClipboard
end;
```

procedure PasteFromClipboard;

Value

None

Purpose

Paste an image from the clipboard into the MultiImage.

Remark

SHIFT INSERT and CTRL V does the same

Example

```
procedure TForm1.Paste1Click(Sender: TObject);
begin
    DBMultiImage1.PasteFromClipboard;
end;
```

Credits Scrolling TBobField Messages

Overview

Credit messages are TDBMultimages created by the VCL on the fly. Stored in the blob are:

MessageFont : TFont; the message's font
MessageSpeed : Integer the scrolling speed 1 is fast 10 is slow
MessageColor : TColor; the background color
CreditBoxList : TStringList; the credit messages in a stringlist

The VCL does NOT have its own moving engine. You "the programmer" must trigger the movements. The reason for this is that an application can have only one Application.OnIdle event. This event needs to be shared with other events which may need an OnIdle event. Note that other VCLs could also use a Trigger. Make sure that their OnIdle proc. doesn't destroy Multimage's trigger.

In your application you need to add a procedure to the private clauses called e.g. Trigger:

```
type
  TForm1 = class(TForm)
private
  Procedure Trigger(Sender : TObject; Var Done : Boolean);
public
```

In the form create you will assign Trigger to the onidle event.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.OnIdle:=Trigger;
end;
```

The procedure trigger will then trigger the VCL:

```
Procedure TForm1.Trigger(Sender : TObject; Var Done : Boolean);
begin
  DBMultImage1.Trigger;
end;
```

Procedure Trigger;

Value

None

Purpose

Trigger the scrolling message movements.

Example

```
Procedure TForm1.Trigger(Sender : TObject; Var Done : Boolean);  
begin  
    DBMultImage1.Trigger;  
end;
```

Function CreateCreditMessage : Boolean;

Purpose

CreateCreditMessage will open the Message editor. The user can create his own Credit message and save this message to a file with a CMS extension as default.

Return

True or false

Example

```
procedure TBtnBottomDlg.BitBtn7Click(Sender: TObject);  
begin  
  
    Table1.Append;  
  
    If DBMultiMedia1.CreateCreditMessage then  
        Table1.Post  
    else  
        Table1.Cancel;  
end;
```

Note: To save current blob message to a file use SaveToFile.

procedure NewCreditMessage:

Value

None

Purpose

Initiate a new message. Ideal to show messages created on the fly.

Example

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
    DBMultImage1.FreeMsg;
    DBMultImage1.CreditBoxList.Clear;
    DBMultImage1.CreditBoxList.Add(' ImageLib');
    DBMultImage1.CreditBoxList.Add(' Another fine product of');
    DBMultImage1.CreditBoxList.Add(' SKYLINE TOOLS');
    DBMultImage1.CreditBoxList.Add(' Programming : Kevin Adams');
    DBMultImage1.CreditBoxList.Add(' Programming : Jan Dekkers');
    DBMultImage1.CreditBoxList.Add(' Artwork & PR: Jillian Pinsker');
    DBMultImage1.MessageFont.Name:='Arial';
    DBMultImage1.MessageFont.Size:=-40;
    DBMultImage1.MessageFont.Style:=[fsitalic, fsbold];
    DBMultImage1.MessageFont.Color:=clWhite;
    DBMultImage1.MessageColor:=clNavy;
    DBMultImage1.MessageSpeed:=1;
    DBMultImage1.NewCreditMessage;
end;
```

Procedure FreeMsg:

Value

None

Purpose

Disposes the current message and assigns the Picture to Nil

Example

```
procedure TForm1.BitBtn5Click(Sender: TObject);
begin
    DBMultImage1.FreeMsg;
end;
```

Scrolling TBobField Messages

Overview

Scrolling messages are TDBMultimages created by the VCL on the fly. The average blob of a Scrolling message is only 200 bytes. Stored in the blob are:

MessageText	: String;	The message text.
MessageFont	: Tfont;	The message font
MessageColor	: Tcolor;	Background color
MessageSpeed	: Integer;	Scrolling Speed

The VCL does NOT have its own moving engine. You "the programmer" must trigger the movements. The reason is that an application can have only one Application.OnIdle event. This event then needs to be shared with other events which may need an application. Note that other VCLs could also use a Trigger. Make sure that their OnIdle proc. doesn't destroy Multimage's trigger.

In your application you need to add a procedure to the private clauses called e.g. Trigger:

```
type
  TForm1 = class(TForm)
private
    Procedure Trigger(Sender : TObject; Var Done : Boolean);
public
```

In the FormCreate you will assign Trigger to the onIdle event.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    Application.OnIdle:=Trigger;
end;
```

The procedure trigger will then trigger the VCL:

```
Procedure TForm1.Trigger(Sender : TObject; Var Done : Boolean);
begin
    DBMultimage1.Trigger;
end;
```

Procedure Trigger;

Value

None

Purpose

Trigger the scrolling message movements.

Example

```
Procedure TForm1.Trigger(Sender : TObject; Var Done : Boolean);  
begin  
    DBMultilImage1.Trigger;  
end;
```

Function CreateMessage : boolean;

Value

None

Purpose

CreateMessage will open the Message editor. The user can create his own scrolling message and store it in the blobfield.

Returns

True if successful otherwise false

Example

```
procedure TForm1.BitBtn13Click(Sender: TObject);  
begin  
    Table1.Append;  
    If DBMultilImage1.CreateMessage then  
        Table1.Post  
    else  
        Table1.Cancel;  
end;
```

Note: To save current blob message to a file use SaveToFile.

procedure NewMessage;

Value

None

Purpose

Initiate a new message. Ideal to show messages created on the fly.

Example

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
    DBMultImage1.MessageText:='ImageLib 3.0 A great tool ';
    DBMultImage1.MessageFont.Name:='Arial';
    DBMultImage1.MessageFont.Size:=-40;
    DBMultImage1.MessageFont.Style:=[fsitalic, fsbold];
    DBMultImage1.MessageFont.Color:=clWhite;
    DBMultImage1.MessageColor:=clNavy;
    DBMultImage1.MessageSpeed:=1;
    DBMultImage1.NewMessage;
end;
```

Procedure FreeMsg;

Value

None

Purpose

Dispose the current message and then assign Picture to Nil

Example

```
procedure TForm1.BitBtn5Click(Sender: TObject);
begin
    DBMultImage1.FreeMsg;
end;
```


TDBMULTIMEDIA and TDBMEDIAPLAYER: Sample project: MMBLOB.dpr

Overview

DBMultiMedia has all the same properties and functions as DBMultiImage. However, besides the storing and displaying of JPEG, BMP, GIF, PNG, CMS, SCM and PCX from a TBLOBField ,it also stores and plays AVI, MOV, MID, WAV and RMI multimedia files. DBMediaPlayer is a derived Delphi MediaPlayer and has exactly all the same functions and properties. When using the DBMediaPlayer you don't need to assign anything to DBMediaPlayer directly, DBMultiMedia will take care of it.

TDBMULTIMEDIA will automatically enable/disable the playback of:

AVI:	If video for windows isn't installed;
MOV:	If quicktime for windows isn't installed;
WAV:	If no sound support is installed;
RMI:	If no midi playback drivers are installed;
MID:	If no midi playback drivers are installed.

Thus you don't need to be afraid of your program crashing when no sound card is installed or Video for windows isn't present.

Again, all the properties from DBMultiImage are there and we added the following:

function GetMultiMediaExtensions : String;

Value

None

Purpose

This function will return all multimedia extensions from the computer running your application and those supported by DBMultiMedia in the filter format used by the filedialog.

Remark

Run the example file MMBLOB.DPR. You will notice that the Append MM dialogbox contains all the Multimedia supported by the VCL and your PC.

Example

```
procedure TBtnBottomDlg.BitBtn1Click(Sender: TObject);
begin
  OpenFileDialog1.filter:=DBMultiMedia1.GetMultiMediaExtensions;
  if OpenFileDialog1.Execute then begin
    Table1.Append;
    DBMultiMedia1.LoadFromFile(OpenDialog1.FileName);
    Table1.Post;
  end;
end;
```

property PathForTempFile : string

Visual Property

Value

PathName

Purpose

TDBMULTIMEDIA saves its AVI, MOV, WAV, MID and RMI blobs to a temporary file before it is played and then deletes the temporary file. The reason for this is that average multimedia blobs are too large in size to be played from memory. Your application might be distributed and executed from a CD. In order to write a temporary file you need to supply a directory and drive.

Remark

CMS, SCM, JPG, PCX, GIF, PNG and BMP Blobs are not written to a temporary file but expanded directly into memory. If directory or drive doesn't exist it defaults to C:\

Example

```
procedure TBtnBottomDlg.FormCreate(Sender: TObject);  
begin  
  DBMultiMedia1.PathForTempFile:='C:\TEMP';  
end;
```

property TempMov : String

Visual Property

Value

Filename

Default

\$\$\$MOV

Purpose

TDBMULTIMEDIA saves its MOV blobs first to a temporary file before it is played and then deletes the temporary file. This property holds the name of the temporary file.

Example DBMULTIMEDIA1.TempMov:='\$TEMP\$.MOV';

Remark

Since the Delphi MultiMediaPlayer is extension sensitive the extension can't be changed.

property TempAVI : String

Visual Property

Value

Filename

Default

\$\$\$.AVI

Purpose

TDBMULTIMEDIA saves its AVI blobs first to a temporary file before it is played and then deletes the temporary file. This property holds the name of the temporary file.

Example DBMULTIMEDIA1.TempAvi:='\$TEMP\$.AVI';

Remark

Since the Delphi MultiMediaPlayer is extension sensitive the extension can't be changed.

property TempWAV : String

Visual Property

Value

Filename

Default

\$\$\$.WAV

Purpose TDBMULTIMEDIA saves its WAV blobs first to a temporary file before it is played and then deletes the temporary file. This property holds the name of the temporary file.

Example DBMULTIMEDIA1.TempWav:='\$TEMP\$.WAV';

Remark

Since the Delphi MultiMediaPlayer is extension sensitive the extension can't be changed.

property TempMID : String

Visual Property

Value

Filename

Default

\$\$\$MID

Purpose

TDBMULTIMEDIA saves its MID blobs first to a temporary file before it is played and then deletes the temporary file. This property holds the name of the temporary file.

Example DBMULTIMEDIA1.TempMID:='\$TEMP\$.MID';

Remark

Since the Delphi MultiMediaPlayer is extension sensitive the extension can't be changed.

property TempRMI : String

Visual Property

Value

Filename

Default

\$\$\$RMI

Purpose

TDBMULTIMEDIA saves its RMI blobs first to a temporary file before it is played and then deletes the temporary file. This property holds the name of the temporary file.

Example DBMULTIMEDIA1.TempRmi:='\$TEMP\$.RMI';

Remark

Since the Delphi MultiMediaPlayer is extension sensitive the extension can't be changed.

property AutoPlayMultiMedia : Boolean;

Visual Property

Value

True or False

Purpose

If AutoPlayMultiMedia and AutoDisplay are True, the control automatically displays new data when the underlying BLOB field changes (such as when moving to a new record). If AutoPlayMultiMedia and AutoDisplay are False, the control will clear whenever the underlying BLOB field changes. To display the data, the user can double-click on the control or select it and press Enter.

Example

```
procedure TBtnBottomDlg.FormCreate(Sender: TObject);
begin
  DBMultiMedia1.AutoPlayMultiMedia:=true;
end;
```

property AutoRePlayMultiMedia : Boolean

Visual Property

Value

True or False

Purpose

If AutoDisplay and AutoPlayMultiMedia are true, then the multimedia is replayed automatically;

Example

```
procedure TBtnBottomDlg.FormCreate(Sender: TObject);
begin
  DBMultiMedia1.AutoRePlayMultiMedia:=true;
end;
```

property AutoHideMediaPlayer : Boolean;

Visual Property

Value

True or False

Purpose

If the blobfield doesn't contain multimedia it will hide the attached MediaPlayer automatically.

Example

```
procedure TBtnBottomDlg.FormCreate(Sender: TObject);
begin
  DBMultiMedia1.AutoHideMediaPlayer:=true;
```

property MediaPlayer:

Visual Property

Value

DBMediaPlayer

Purpose

ImageLib comes with its own DBmediaplayer directly derived from Tmediaplayer. You need to drop one on your form and set the property MediaPlayer to, for instance: DBmediaplayer1.

Remark

There is no need to attach a filename to DBMediaPlayer. AutoOpen must be false since DBMultiMedia will take care of opening and closing the DBMediaPlayer.

Example

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    DBMultiMedia1.MediaPlayer:=DBMediaPlayer1;
end;
```

DBMediaPlayer1.Display and DisplayRect.**Remark**

In order to display the video in the exact rectangle of your DBMultiMedia you'll need to supply a display and **rect** to the DBMediaPlayer.

Example

```
procedure TBtnBottomDlg.DataSource1DataChange(Sender: TObject; Field: TField);
begin
    DBMediaPlayer1.DisplayRect:=Rect(0,0,DBMultiMedia1.Width,
                                     DBMultiMedia1.Height);
    DBMediaPlayer1.Display:=DBMultiMedia1;
end;
```

PASCAL AND DELPHI DLL Calls and Scrolling messages File/Stream calls

You might never have a need to make calls directly to the DLL. But in case you have a need for it, we listed all the pascal interface callswith the DLL. You can find all the calls in DLL30.INT or DLL30.PAS

We would just like to say a few words about Turbopower. We've used Turbopowers' products for over 4 years now and are very impressed with their "state of the art" development libraries. Their technical support is the best we've ever experienced. They provide a good example for us of how to do business and how to treat customers.

Turbopower's products:

Async Professional,
B-Tree Filer,
Object Professional,
TSRs and more,
Turbo Analyst,
Turbo Professional,
Data Entry Workshop,
Win/Sys Library, and their latest great Delphi product,
Orpheus.

on CompuServe, Go PCVENB to download their free trial libraries.

Contacting TurboPower Sales

Telephone : 800-333-4160 (sales in the U.S. & Canada)
719-260-9136 (international sales)
719-260-7151 (fax)
CompuServe : 76004,2611
Internet : 76004.2611@compuserve.com
Postal mail : TurboPower Software
P.O. Box 49009
Colorado Springs, CO 80949-9009

Gif and Tiff uses LZW compression which is patented by Unisys. On CompuServe GO PICS to obtain information about the Unisys patents. By using ImageLib's GIF Read and Write features you acknowledge that SkyLine has notified you about the LZW patend and hold SkyLine harmless from any legal actions. This work "JPEG file i/o" is based in part on the Independent JPEG Group