A PROGRAMMERS INTRODUCTION TO USING THE IMAGEEN IMAGE LIBRARY

IMAGEEN LIBRARY SUITE

A Programmers Introduction To Using The ImageEn Image Library

Table of Contents

Forward

By Fabrizio Di Vittorio or Nigel Cross

Introduction

ImageEn Image Library

This book is intended for Novice to mid-level Pascal developers to learn how to develop for the ImageEn Image Library. The author began using ImageEN in 1998 and was one of the first users of ImageEn. Each year the author has contributed to the development of ImageEn by positing demos, answering web questions which led to the development of this book.

There are many possible topics that could be discussed in this book but only topics that offer new information or maybe the least understood are presented along with a greatly expanded and proof read help file biased on the English language. Topics such as the database ImageEn components are not discussed because they are seldome used by the author, not because they are not good, but because the author seldom does database programming.

This document contains tips, suggestions and documentation for version 4.0.0.1. Much of the documentation was edited and spell checked with additional examples provided for many functions and procedures. All of the demos were compiled and tested with Delphi 2010.

ImageEn 4.xx

Imaging Components

www.imageen.com

Chapter 1

Getting Started With ImageEn

Chapter 1. Getting Started With ImageEn

"ImageEn is the best Imaging Library for Embaracdero's Delphi VCL."

ImageEn is the most powerful native image library available for Delphi and C++ Builder, and is also available for .NET. The library includes a complete suite of components to handle all aspects of image and editing, analysis and display. Join thousands of other developers who rely on ImageEn to add professional multimedia functionality to their software.

ImageEn supports Delphi 5 - XE2, C++ Builder 5 - XE2 and any programming environment supporting ActiveX objects or .NET 2.0 - 4.0.

ImageEn provides simple components, classes and procedures to accomplish most anything with Images.

File Operations

Getting Started With ImageEN

ImageEn provides support for asynchronous loading and saving and Lossless JPEG rotation and cropping. ImageEn can load and save digital camera (EXIF) fields in JPEG, TIFF, RAW, PSD and HD Photo files (without modifying the original image) as well as load and save of EXIF GPS fields.

ImageEn provides support for Color Management Systems to process ICC profiles.

Loading and saving of Adobe XMP fields from JPEG, TIFF, HD Photo and PSD files is supported. Raw camera support including access to internal thumbnails and other data fields is supported. Loading and saving of IPTC data (most commonly used by Photoshop) from JPEG and TIFF files (without modifying the original image)

ImageEn can quickly read image properties (dimensions, color depth…) without loading the image. Image load and save dialogs which include a preview and relevant save settings is provided.

The alpha channel is supported with GIF, TIFF, PNG, ICO, CUR, TGA and, PSD files. Alpha channel (transparency) and multiple layers (with 37 layer blend modes) is provided. All layers can be moved, resized and rotated (programmatically or by the user)

Images may be loaded directly from the internet using http/ftp protocol. Encryption and decryption is supported using a 128 bit key.

Image Display

Images can be automatically displayed “To-Fit” or with real time zoom. When zooming six quality settings are available to enhance its display (from fastest to best quality) .

Images from digital cameras that support EXIF rotation data can be automatically displayed with the correct orientation. Over 180 stunning image transition and Pan-Zoom effects is provided.

Many selection or rubberbanding types including rectangle, ellipse, polygon, lasso and “magic wand” (to instantly select a colored area) and other options (including selection intensity and feathering) is provided. Selection functions are easy to use and are moveable and resizeable.

Extensive image processing procedures such as cut, Copy, Paste, Crop, Flipping, Rotating, Tinting, Brightness, Contrast, Brighten, Darken, Intensity, Contrast, Saturation, Sharpen, Gradients, SoftShadow, InsideShadow, Grayscale, Negative, Convert, Resample, Resize and AutoEnhance are provided.

A grid may be displayed that zooms with the image that allows pixel level editing.

Other mouse interaction options including mouse wheel support and click-dragging to navigate and zooming or auto fitting the image is provided.

ImageEn has multilevel undo and redo methods. Automatic as well as manual undo methods are provided.

Coverflow-style animation is supported to display and navigate images.

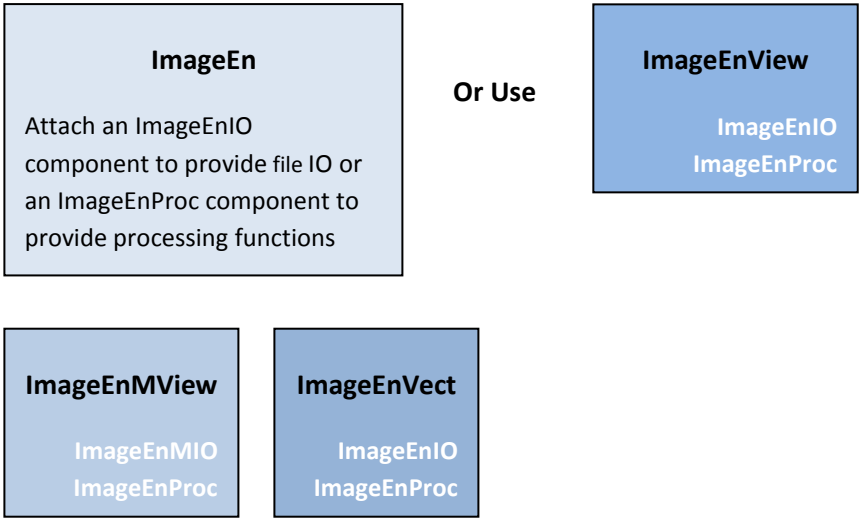
The ImageEn VCL offers:

Getting Started With ImageEN

ImageEn includes 15 components and many classes to assist you with the majority of your imaging needs.

Components

- **ImageEn** - is an image component container and viewer without internal file IO or processing support.
- **ImageEnView** - is an image component container and viewer with built-in file IO and image processing support.
- **ImageEnMView**- is a component similar to TImageEnView, but it can handle multiple images.
- **ImageEnVect** - inherits from TImageEnView (has all method and properties), and handles vectorial objects and measurements.



It is **not** necessary to place an ImageEnIO or ImageEnProc component on the form when using ImageEnView or the ImageEnVect component. To get file IO or image processing methods with ImageEnView or ImageEnVect just add ImageEnIO and ImageEnProc to the uses clause, then use the "built in" ImageEnView1.IO and ImageEnView1.Proc functions. To get file IO methods with ImageEnMView add ImageEnMIO to uses clause, then use the "built-in" ImageEnMView1.MIO functions.

- **ImageEnVideoView** –
- **ImageEnVideoCap**- is a non-visual component which can capture images from video cameras.
- **ImageEnIO** – is a nonvisual component that handles input/output operations on attached TImageEn, TImageEnView, TImageEnDBView, TImageEn, TIEBitmap, TImage or TBitmap components.
- **ImageEnMIO** – is a nonvisual component that handles input/output operations on a attached TImageEnMView component.
- **ImageEnProc**- is a nonvisual component that provides image processing and analysis tasks on attached TImageEn, TImageEnView, TImageEnDBView, TImageEn, TIEBitmap, TImage or TBitmap components.
- **HistogramBox**- displays the color channels histogram
- **Gradient Bar-** shows a gradient bar, with a grip with which the user can select a color or an index.
- **HSVBox-** allows selection of a color in the HSV (Hue Saturation Value) color space.
- **OpenImageEnDialog-** displays a modal Windows dialog box for selecting and opening graphics files.
- **SaveImageEnDialog-** is the same as the TOpenImageEnDialog, but with "Advanced" button to display and modify advanced specific file format parameters.
- **Rulerbox-** is a ruler. It was included in ImageEn for the histogram equalization preview dialog.

Classes

ImageEn includes nearly 20 classes to assist you with the majority of your imaging needs.

- **TIEHashStream**- Builds a hash string from stream.
- **TIEResourceExtractor** class helps to extract resources from a PE files like EXE, DLL, OCX, ICL, BPL
- **TIETIFFHandler** class allows you to read all TIFF tags, edit/add tags and pages, delete pages, and finally save back the modified TIFF.
- **TIETagsHandler**-This class allows reading EXIF maker note tags, when it is in form of IFD (like Canon).
- **TIEWICReader** class encapsulates some Microsoft Windows Imaging Component (WIC) interfaces and allows loading Microsoft HD Photo and other WIC installed file formats.
- **TIEWICWriter** class encapsulates some Microsoft Windows Imaging Component (WIC) interfaces and allows writing Microsoft HD Photo and other WIC installed file formats.
- **TIOParamsVals** object contains all parameters for all image file formats handled by ImageEn, such as compression type, text comments, bit per sample, samples per pixel, etc.
- **TIERFBClient** implements a RFB (Remote Frame Buffer) client. It can connect to any RFB compatible server like RealVNC, TightVNC, VMWare virtual machines or Macintosh remote desktop.
- **TIEDirectShow** class allows control of some Direct Show features, such as video capture, audio capture; multimedia files capture as well video rendering, and multimedia file writing.
- **TIEWia** class (TImageEnIO.WIAParams and TImageEnMIO.WIAParams properties) allows you to set/get parameters, show dialogs and control WIA devices.
- **TIEMediaReader** is a simple stand-alone class to get frames from a media file (wmv, mpeg...)., with one single method to copy the wanted frame to a TIEBitmap object.
- **TIETWainParams** contains many properties and methods to control TWain scanners without using the default dialog.

- **TIEBitmap** is a replacement of VCL TBitmap class. It has many methods and properties compatible with TBitmap and enhances it supporting multi-threading and large images. TIEBitmap can store images in memory mapped files (for big images), in memory (fast access) or can encapsulate TBitmap objects (for canvas drawing and compatibility).
- **TIEMask** contains a selection, which is a map of selected and unselected pixels. A selection map can have a depth of 1 bit or 8 bit. For a map of 1 bit, 0 is a non-selected pixel, while 1 is selected. For a map of 8 bit, 0 is non-selected pixel, >0 is "semi" selected pixel up to 255 that means fully selected pixel.
- **TIEAnimation** is the base abstract class for animation classes like coverflow-like (TIEHorizontalFlow) and circular cflow (TIECircularFlow) or any user custom animation classes).
- **TIEHorizontalFlow**- Implements a coverflow-like animation.
- **TIECircularFlow**- Implements a circular animation.
- **TIEImageList** – is a simple, but powerful in memory list of TIEBitmap images.

Image Editing and Processing

Image resizing with eleven quality filters including Triangle, Hermite, Bell, BSpline, Lanczos3, Mitchell, FastLinear, Bilinear and Bicubic is provided.

Many color adjustment facilities including contrast, HSL, HSV channel separation, RGB, histogram equalization, Fast Fourier Transformation (FFT), gamma correction, temperature and noise removal is provided. Color adjustment facilities can be applied programmatically or using a built-in dialog. ImageEn supports effects including custom filters, bump map, lens, wave, morphing, gaussian and motion blurring and sharpening. All effects can be applied programmatically or using a built-in dialog.

ImageEn can create grayscale and negative images and can convert color ranges

Image cropping, auto-cropping, and flipping or rotation with anti-alias and edge and skew detection support is provided along with red-eye removal and soft shadow and inner shadow effects.

ImageEn supports a wide range of native pixel formats including 1 bit, 8 bit paletted, 8 bit grayscale, 16 bit grayscale, 24 bit RGB, 32 bit float point, 24 bit CMYK, 48 bit RGB and CIELab.

Image and Video Acquisition

ImageEn can acquire images from twain (modal and modeless) and WIA compatible scanners and cameras or capture from the screen.

Visual and non-visual components for video capture (supporting freeze frames and real-time processing) are provided. ImageEn supports all installed codecs and video capture cards. Video capture and saving of multimedia using DirectShow is provided.

Thumbnails

A powerful thumbnail component that displays a grid of images by reading from a folder or database table is provided along with many style and other customizations, including wallpaper. ImageEn supports multiple selections; display all frames/pages of AVI, GIF, TIFF or video files using Direct Show. ImageEn is very memory efficient and fast with multi-threading background image loading, caching and optional use of embedded thumbnails.

Vectorial Object Component

A vectorial editing component that allows you to add and manipulate objects on a background image is provided. The vectorial component offers object support for lines, boxes, circles, ellipses, bitmaps, text (including curved, multi-line and formatted), rulers, polylines, polygons, angles and arrows. Many object options' including transparency, soft-shadow and anti-alias is provided. Objects may be saved or loaded to compressed IEV format or imported from AutoCAD DXF files. The vectorial editor can measure lines, perimeters, areas and angles and like the image editor has multi-level undo and redo support and provides full clipboard support.

Other Features

Other features included with ImageEn include printing of single images and sheets of multiple images complete with print preview support.

Data-aware versions of image and thumbnail components are provided to automatically display files stored as blob or path references in a database table.

One-click selection of languages for all dialogs with support for: English, Italian, German, Spanish, French, Portuguese, Greek, Russian, Dutch, Swedish, Polish, Japanese, Czech, Finnish, Farsi, Chinese, Danish, Turkish and Hungarian

ImageEn file input and output and image processing also works with the standard TImage component and TPicture class or with bitmaps.

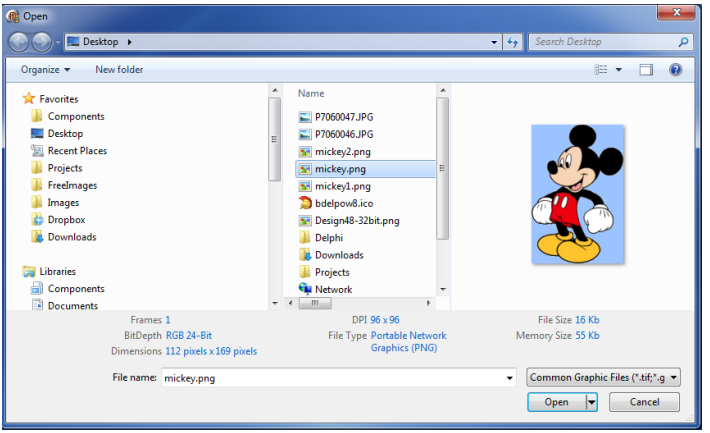
Supported File Formats

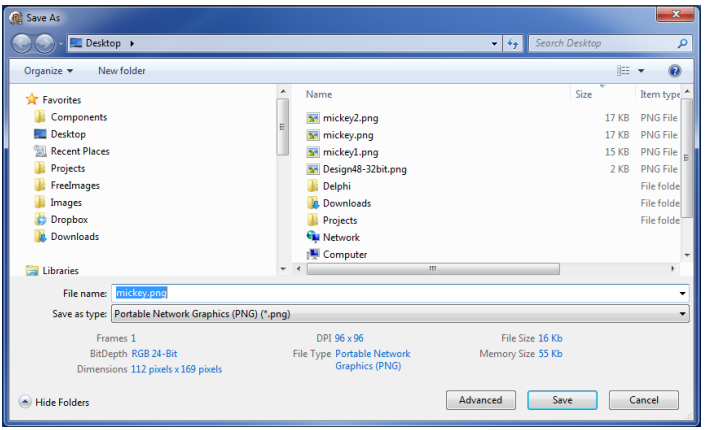
Format	Notes	Load	Save
JPEG	Supports ½, ¼ and 1/8 sub-sizes for fast preview	✓	✓
JPEG2000		✓	✓
GIF	Including editing and display of animated GIFs	✓	✓
PNG		✓	✓
BMP	Compressed and uncompressed	✓	✓
TIFF	Editing and display of single and multipage TIFF. Also supports FAX, G3F and G3N sub-formats	✓	✓
PCX	Including DCX (Multipage PCX) format	✓	✓
Raw Camera Formats	Including Digital Negative Format (*.dng), Canon (*.cr2, *.crw), Kodak (*.dcr), Minolta (*.mrw), Nikon (*.nef), Olympus (*.orf), Pentax (*.pef), Fuji (*.raf), Leica (*.raw), Sony (*.srf) and Sigma (*.x3f)	✓	✓
Icons (ICO)	With multiple resolution and color depth support	✓	✓
DICOM (Medical Imaging)	Single and multipage	✓	
Adobe	With multiple layer support	✓	✓

Format	Notes	Load	Save
Photoshop (PSD)			
WMP	Also known as Microsoft HD Photo	✓	✓
PostScript (PS and EPS)	Single and multipage		✓
Adobe PDF	Single and multipage images		✓
Metafiles (WMF and EMF)		✓	
Cursors (CUR)		✓	
AVI	Including retrieval and modification of frames	✓	✓
Media formats (MPEG, WMV, etc.)	Via the DirectShow API	✓	✓

Modern File Open and Save Dialogs

The TIEImageENOpenDialog and TIEImageENSaveDialog offer some useful features over the TOpenPicture and TSavePicture dialogs provided with Delphi. Unfortunately the ImageEn dialogs use an old interface so to be consistent with today's operating system I customized Vista/Windows 7 FileDialogs.





To customize Vista's and Windows 7 common file dialog, MS introduced the IFileDialogCustomize interface. Previous to Vista, the customization was done by a custom dialog template resource. This is how the IEOpenDialog and IESaveDialog components were built.

The old customization method will not work under Vista unless the file dialog is restricted to XP's style. There is no way to make the old template method work with the Vista's native dialog. The only way is to make use of the IFileDialogCustomize interface. The major drawback of the new method is the extraordinary limitation/restriction of supported control types. Firstly, the IFileDialogCustomize interface only works with Vista or above. Second, the choice of controls are very limited and **no user-defined controls are allowed** so an TImageEnView cannot be added to the dialogs.

The controls provided by IFileDialogCustomize are Menu, Button, Combo box, Radio button list, Check button (check box), Edit, Separator and Label.

TImageEnView cannot be used for the preview so we are stuck with the dialogs built-in preview for the time being, but we can add labels to display image information as well as add an "Advanced" button to call TImageENIO.DoPreviews. As a final step I believe ImageEnIO file support will be added to the dialogs eventually as well.

Delphi 2009 introduced a new TFileOpenDialog component for Vista. The new class implements the IFileDialog interface. The IFileDialogCustomize can be queried from the TFileOpenDialog.Dialog property. Note that the TFileOpenDialog.Dialog is always nil unless the dialog is being shown. To get the IFileDialogCustomize interface, we do some coding inside the TFileOpenDialog.OnExecute() or TFileSaveDialog.OnExecute() events.

The added controls are pretty useless unless you can interact with them but we only need to add support for a PushButton event for the SaveDialogs Advanced button. I have implemented this as well. There are IFileDialogEvents and IFileDialogControlEvents interfaces to deal with the controls. To implement it, simply declare a class: TMyFileDialogEvents = class(TInterfacedObject, IFileDialogEvents, IFileDialogControlEvents) and provide the needed methods.

*See the **Example** in TFileDialogsWithImageEn.zip*

To add labels to the TFileOpenDialog for displaying image information:

```
procedure TForm1.FileOpenDialog1Execute(Sender:
TObject);
const
  dwVisualGroup1ID: DWORD = 1200;
  dwVisualGroup2ID: DWORD = 1300;
  dwVisualGroup3ID: DWORD = 1400;
  dwVisualGroup4ID: DWORD = 1500;
  dwVisualGroup5ID: DWORD = 1600;
  dwVisualGroup6ID: DWORD = 1700;
  dwVisualGroup7ID: DWORD = 1800;
var
  c: IFileDialogCustomize;
begin
  if FileOpenDialog1.Dialog.QueryInterface(
    IFileDialogCustomize, c) = S_OK then
  begin
    // Add a visual group for Frames
    c.StartVisualGroup(dwVisualGroup1ID,
      'Frames');
    c.AddControlItem(1, 0, 'Frames: ');
    c.AddText(1, '');
    c.EndVisualGroup;
    // Add a visual group for BitDepth
    c.StartVisualGroup(dwVisualGroup2ID,
      'BitDepth');
    c.AddControlItem(2, 1, 'BitDepth: ');
    c.AddText(2, '');
    c.EndVisualGroup;
    // Add a visual group for Dimensions
    c.StartVisualGroup(dwVisualGroup3ID,
      'Dimensions');
    c.AddControlItem(3, 2, 'Dimensions:');
    c.AddText(3, '');
```

```
    c.EndVisualGroup;
    // Add a visual group for DPI
    c.StartVisualGroup(dwVisualGroup4ID, 'DPI');
    c.AddControlItem(4, 3, 'DPI:');
    c.AddText(4, '');
    c.EndVisualGroup;
    // Add a visual group for FileType
    c.StartVisualGroup(dwVisualGroup5ID, 'File
      Type');
    c.AddControlItem(5, 4, 'File Type:');
    c.AddText(5, '');
    c.EndVisualGroup;
    // Add a visual group for FileSize
    c.StartVisualGroup(dwVisualGroup6ID, 'File
      Size');
    c.AddControlItem(6, 5, 'File Type:');
    c.AddText(6, '');
    c.EndVisualGroup;
    // Add a visual group for MemorySize
    c.StartVisualGroup(dwVisualGroup7ID, 'Memory
      Size');
    c.AddControlItem(7, 6, 'Memory:');
    c.AddText(7, '');
    c.EndVisualGroup;
  end;
end;
```

To display the image information for the selected image:

```
procedure
TForm1.FileOpenDialog1SelectionChange(Sender:
   TObject);
var
  c: IFileDialogCustomize;
  iFilename: string;
```

```
  iFrames: integer;
  iBitDepth: integer;
  iBitCount: integer;
  idim: integer;
  iColors: string;
  iDimensions: string;
  iDPI: string;
  iFileType: string;
  iFileSize: string;
  fFileSize: integer;
  iMemorySize: string;
  iImageENIO: TImageENIO;
begin
  if FileOpenDialog1.Dialog.QueryInterface(
    IFileDialogCustomize, c) = S_OK then
  begin
    // Get the Filename
    iFileName := FileOpenDialog1.FileName;
    if FileExists(iFileName) then
    begin
      // Create a ImageENIO to get the
      // ParamsFromFile
      iImageENIO := TImageENIO.Create(nil);
      try
        // Get the Params
        iImageENIO.ParamsFromFile(iFileName);
        // Get the BitDepth
        iBitDepth :=
          iImageENIO.Params.BitsPerSample *
            iImageENIO.Params.SamplesPerPixel;
```

```
case iBitDepth of
  32: iColors := 'RGBA 32-Bit';
else
  iColors := 'RGB ' + IntToStr(iBitDepth)
             + '-Bit';
end;
// Get the Dimensions
iDimensions := IntegerToString(
               iImageENIO. Params.
               Width) + ' pixels x ' +
               IntegerToString(iImageENIO
               .Params.Height) + '
                pixels';
// Get the DPI
iDPI := IntToStr(iImageENIO.Params.DpiX)
         + ' x ' +
        IntToStr(iImageENIO.Params.DpiY);

// Get the FileTypeStr
iFileType :=
           iImageENIO.Params.FileTypeStr;
// Get the FileSize
fFileSize := IEGetFileSize(iFileName);
if fFilesize <> -1 then
  if fFileSize < 1024 then
    iFileSize := IntToStr(fFileSize) +
                 ' bytes'
  else
    iFileSize := IntToStr(fFileSize div
                 1024) + ' Kb';
```

```
// Set the BitCount
if (iImageENIO.Params.SamplesPerPixel =
  1) and (iImageENIO.Params.
  BitsPerSample = 1) then
  iBitCount := 1
else
  iBitCount := 24;
// Get the MemorySize
iFrames := 1;
idim := (((iImageENIO.Params.Width *
        iBitCount) + 31) div 32) * 4 *
        iImageENIO.Params.Height *
        iFrames;
if idim < 1024 then
  iMemorySize := IntToStr(idim) +
                 ' bytes'
else
  iMemorySize := IntToStr(idim div 1024)
                 + ' Kb';
// Get the number of frames
iFrames := IEGetFileFramesCount(
           iFileName);
// Set the FileOpenDialog labels
c.SetControlLabel(1, PWideChar(IntToStr(
                 iFrames)));
c.SetControlLabel(2, PWideChar(iColors));
c.SetControlLabel(3, PWideChar(
  iDimensions));
c.SetControlLabel(4, PWideChar(iDPI));
c.SetControlLabel(5, PWideChar(
  iFileType));
```

```
        c.SetControlLabel(6, PWideChar(
          iFileSize));
        c.SetControlLabel(7,
          PWideChar(iMemorySize));
      finally
        iImageENIO.Free;
      end;
    end
    else
    begin
      if DirectoryExists(iFileName) then
      begin
        FileOpenDialog1.Dialog.SetFolder(
          FileOpenDialog1.ShellItem);
      end;
    end;
  end;
end;
```

To set up the file filter:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  // Check Operating System for vista or windows
  // 7 and abort if not either
  if not IsVista then
    if not IsWindows7 then
    begin
```

```
      ShowMessage('This demonstrates how to use
        the new Vista or Windows7 common file
        dialogs and requires either Vista or
        Windows 7.');
      Application.Terminate;
    end;
  // Add the filetypes to TFileOpenDialog
  with FileOpenDialog1.FileTypes.Add do
  begin
    DisplayName := 'All Files (*.*)';
    FileMask := '*.*';
  end;
  with FileOpenDialog1.FileTypes.Add do
  begin
    DisplayName := 'Common Graphic Files';
    FileMask :=
'*.tif;*.gif;*.png;*.png;*.apf;*.cur;*.pcx;*.ani;
*.gif;*.png;*.jpg;*.jpeg;*.bmp;*.tif;*.tiff;*.ico
;*.emf;*.wmf';
  end;
  with FileOpenDialog1.FileTypes.Add do
  begin
    DisplayName := 'JPEG Bitmap
(JPG;JPEG;JPE;JIF)';
    FileMask := '*.jpg;*.jpeg;*.jpe;*.jif';
  end;
  with FileOpenDialog1.FileTypes.Add do
  begin
    DisplayName := 'TIFF Bitmap
                    (TIF;TIFF;FAX;G2N;G3F;XIF)';
    FileMask := *.tif;*.tiff;*.fax;*.g3n;*.g3f;
                *.xif';
  end;
```

```
with FileOpenDialog1.FileTypes.Add do
begin
  DisplayName := 'CompuServe Bitmap (GIF)';
  FileMask := '*.gif';
end;
with FileOpenDialog1.FileTypes.Add do
begin
  DisplayName := 'PaintBrush (PCX)';
  FileMask := '*.pcx';
end;
with FileOpenDialog1.FileTypes.Add do
begin
  DisplayName := 'Windows Bitmap
                 (BMP;DIB;RLE)';
  FileMask := '*.bmp;*.dib;*.rle';
end;
with FileOpenDialog1.FileTypes.Add do
begin
  DisplayName := 'Windows Icon (ICO)';
  FileMask := '*.ico';
end;
with FileOpenDialog1.FileTypes.Add do
begin
  DisplayName := 'Portable Network Graphics
                 (PNG)';
  FileMask := '*.png';
end;
with FileOpenDialog1.FileTypes.Add do
begin
  DisplayName := 'DICOM Bitmap
                 (DCM;DIC;DICOM;V2)';
  FileMask := '*.dcm;*.dic;*.dicom;*.v2';
end;
```

```
with FileOpenDialog1.FileTypes.Add do
begin
  DisplayName := 'Windows Metafile (WMF)';
  FileMask := '*.wmf';
end;
with FileOpenDialog1.FileTypes.Add do
begin
  DisplayName := 'Targa Bitmap
                 (TGA;TARGA;VDA;ICB;VST;PIX)';
  FileMask := '*.tga;*.targa;*.vda;*.icb;*.vst;
              *.pix';
end;
with FileOpenDialog1.FileTypes.Add do
begin
  DisplayName := 'Portable Pixmap, GrayMap,
                  BitMap (PXM;PPM;PGM;PBM)';
  FileMask := '*.pxm;*.ppm;*.pgm;*.pbm';
end;
with FileOpenDialog1.FileTypes.Add do
begin
  DisplayName := 'Wireless Bitmap (WBMP)';
  FileMask := '*.wbmp';
end;
with FileOpenDialog1.FileTypes.Add do  begin
  DisplayName := 'JPEG2000 (JP2)';
  FileMask := '*.jp2';
end;
```

```
  with FileOpenDialog1.FileTypes.Add do
  begin
    DisplayName := 'JPEG2000 Code Stream
                   (J2K;JPC;J2C)';
    FileMask := '*.j2k;*.jpc;*.j2c';
  end;
  with FileOpenDialog1.FileTypes.Add do
  begin
    DisplayName := 'Multipage PCX (DCX)';
    FileMask := '*.dcx';
  end;
  with FileOpenDialog1.FileTypes.Add do
  begin
    DisplayName := 'Camera RAW (CRW; CR2; EF;
                    RAW; PEF; RAF; X3F; BAY; ORF;
                    SRF; MRW; DCR; SR2)';
    FileMask := '*.crw; *.cr2; *.nef; *.raw;
                *.pef; *.raf; *.x3f; *.bay;
                *.orf; *.srf; *.mrw; *.dcr;
                *.sr2';
  end;
  with FileOpenDialog1.FileTypes.Add do
  begin
    DisplayName := 'Photoshop PSD (PSD)';
    FileMask := '*.psd';
  end;
  with FileOpenDialog1.FileTypes.Add do
  begin
    DisplayName := 'Microsoft HD Photo
                   (WDP;HDP)';
    FileMask := '*.wdp;*.hdp';
  end;
end;
```

To add labels to the TFileSaveDialog for displaying image information:

```
procedure TForm1.FileSaveDialog1Execute(Sender:
TObject);
const
  dwVisualGroup1ID: DWORD = 1200;
  dwVisualGroup2ID: DWORD = 1300;
  dwVisualGroup3ID: DWORD = 1400;
  dwVisualGroup4ID: DWORD = 1500;
  dwVisualGroup5ID: DWORD = 1600;
  dwVisualGroup6ID: DWORD = 1700;
  dwVisualGroup7ID: DWORD = 1800;
var
  c: IFileDialogCustomize;
begin
  if Supports(FileSaveDialog1.Dialog,
    IFileDialogCustomize, c) then
  begin
    // Add a visual group for Frames
    c.StartVisualGroup(dwVisualGroup1ID,
     'Frames');
    c.AddControlItem(1, 0, 'Frames: ');
    c.AddText(1, '');
    c.EndVisualGroup;
    // Add a visual group for BitDepth
    c.StartVisualGroup(dwVisualGroup2ID,
      'BitDepth');
    c.AddControlItem(2, 1, 'BitDepth: ');
    c.AddText(2, '');
    c.EndVisualGroup;
```

```
    // Add a visual group for Dimensions
    c.StartVisualGroup(dwVisualGroup3ID,
      'Dimensions');
    c.AddControlItem(3, 2, 'Dimensions:');
    c.AddText(3, '');
    c.EndVisualGroup;
    // Add a visual group for DPI
    c.StartVisualGroup(dwVisualGroup4ID, 'DPI');
    c.AddControlItem(4, 3, 'DPI:');
    c.AddText(4, '');
    c.EndVisualGroup;
    // Add a visual group for FileType
    c.StartVisualGroup(dwVisualGroup5ID, 'File
      Type');
    c.AddControlItem(5, 4, 'File Type:');
    c.AddText(5, '');
    c.EndVisualGroup;
    // Add a visual group for FileSize
    c.StartVisualGroup(dwVisualGroup6ID, 'File
      Size');
    c.AddControlItem(6, 5, 'File Type:');
    c.AddText(6, '');
    c.EndVisualGroup;
    // Add a visual group for MemorySize
    c.StartVisualGroup(dwVisualGroup7ID, 'Memory
      Size');
    c.AddControlItem(7, 6, 'Memory:');
    c.AddText(7, '');
    c.EndVisualGroup;
  end;
end;
```

To display the image information for the selected image:

```
procedure TForm1.FileSaveDialog1SelectionChange(
  Sender: TObject);
var
  c: IFileDialogCustomize;
  iFilename: string;
  iFrames: integer;
  iBitDepth: integer;
  iBitCount: integer;
  idim: integer;
  iColors: string;
  iDimensions: string;
  iDPI: string;
  iFileType: string;
  iFileSize: string;
  fFileSize: integer;
  iMemorySize: string;
  iImageENIO: TImageENIO;
begin
  if FileSaveDialog1.Dialog.QueryInterface(
    IFileDialogCustomize, c) = S_OK then
     begin
       // Get the Filename
       iFileName := FileSaveDialog1.FileName;
       if FileExists(iFileName) then
       begin
         // Create a ImageENIO to get the
         ParamsFromFile
         iImageENIO := TImageENIO.Create(nil);
         try
            // Get the Params
            iImageENIO.ParamsFromFile(iFileName);
          // Get the BitDepth
```

```
iBitDepth := iImageENIO.Params.
             BitsPerSample *
             ImageENIO.Params.
             SamplesPerPixel;
case iBitDepth of
  32: iColors := 'RGBA 32-Bit';
else
  iColors := 'RGB ' + IntToStr(iBitDepth)
             + '-Bit';
end;
// Get the Dimensions
iDimensions := IntegerToString(
               iImageENIO. Params. Width)
               + ' pixels x ' +
               IntegerToString(
               iImageENIO.Params.
               Height) + ' pixels';
// Get the DPI
iDPI := IntToStr(iImageENIO.Params.DpiX)
        + ' x ' + IntToStr(iImageENIO.
        Params.DpiY);
// Get the FileTypeStr
iFileType := iImageENIO.Params.
             FileTypeStr;
// Get the FileSize
fFileSize := IEGetFileSize(iFileName);
if fFilesize <> -1 then
  if fFileSize < 1024 then
    iFileSize := IntToStr(fFileSize) + '
                 bytes'
  else
    iFileSize := IntToStr(fFileSize div
                 1024) + ' Kb';
    // Set the BitCount
```

```
if (iImageENIO.Params.SamplesPerPixel =
   1) and (iImageENIO.Params.
   BitsPerSample = 1) then
      iBitCount := 1
else
  iBitCount := 24;
// Get the MemorySize
iFrames := 1;
idim := (((iImageENIO.Params.Width
        * iBitCount) + 31) div 32) * 4 *
         iImageENIO.Params.Height *
        iFrames;
if idim < 1024 then
  iMemorySize := IntToStr(idim) + '
                 bytes'
else
  iMemorySize := IntToStr(idim div 1024)
                  + ' Kb';
// Get the number of frames
iFrames := IEGetFileFramesCount(
           iFileName);
// Set the FileOpenDialog labels
c.SetControlLabel(1, PWideChar(
   IntToStr(iFrames)));
c.SetControlLabel(2, PWideChar(iColors));
c.SetControlLabel(3, PWideChar(
   iDimensions));
c.SetControlLabel(4, PWideChar(iDPI));
c.SetControlLabel(5, PWideChar(
   iFileType));
c.SetControlLabel(6, PWideChar(
   iFileSize));
c.SetControlLabel(7, PWideChar(
   iMemorySize));
```

```
      finally
        iImageENIO.Free;
      end;
    end;
  end;
end;
```

To change the file type when the filter changes:

```
procedure
TForm1.FileSaveDialog1TypeChange(Sender:
    TObject);
// Change the FileSaveDialog filename when the
   FileType changes
var
  iFileName: string;
  iExt: string;
  iName: PChar;
begin
  with TFileSaveDialog(Sender) do
  begin
    Dialog.GetFileName(iName);
    if iName = '' then // FileName is Empty
      exit;
    case FileTypeIndex of
      1: iExt := '.jpg';
      2: iExt := '.jpeg';
      3: iExt := '.gif';
      4: iExt := '.tif';
      5: iExt := '.tiff';
      6: iExt := '.pcx';
      7: iExt := '.png';
```

```
      8: iExt := '.ico';
      9: iExt := '.bmp';
      10: iExt := '.tga';
      11: iExt := '.pxm';
      12: iExt := '.ppm';
      13: iExt := '.pgm';
      14: iExt := '.pbm';
      15: iExt := '.jp2';
      16: iExt := '.j2k';
      17: iExt := '.dcx';
      18: iExt := '.wdp';
      19: iExt := '.hdp';
    end;
    iFileName := ChangeFileExt(ExtractFileName
                 (iName), iExt);
    Dialog.SetFileName(PChar(iFileName));
  end;
end;
```

Detect a "Advanced Button Click in the TFileSaveDialog:

```
type  TMyFileDialogEvents = class(
    TInterfacedObject, IFileDialogEvents,
    IFileDialogControlEvents)
public
  { IFileDialogEvents }
  function OnFileOk(const pfd: IFileDialog): HResult; stdcall;
  function OnFolderChanging(const pfd: IFileDialog; const psiFolder:
IShellItem): HResult; stdcall;
  function OnFolderChange(const pfd: IFileDialog): HResult; stdcall;
  function OnSelectionChange(const pfd: IFileDialog): HResult; stdcall;
  function OnShareViolation(const pfd: IFileDialog; const psi:
IShellItem;
    out pResponse: DWORD): HResult; stdcall;
  function OnTypeChange(const pfd: IFileDialog): HResult; stdcall;
  function OnOverwrite(const pfd: IFileDialog; const psi: IShellItem; out
pResponse: DWORD): HResult; stdcall;
  { IFileDialogControlEvents }
  function OnItemSelected(const pfdc: IFileDialogCustomize; dwIDCtl:
DWORD;
    dwIDItem: DWORD): HResult; stdcall;
  function OnButtonClicked(const pfdc: IFileDialogCustomize;
    dwIDCtl: DWORD): HResult; stdcall;
  function OnCheckButtonToggled(const pfdc: IFileDialogCustomize;
    dwIDCtl: DWORD; bChecked: BOOL): HResult; stdcall;
```

```
    function OnControlActivating(const pfdc:
IFileDialogCustomize;
      dwIDCtl: DWORD): HResult; stdcall;
  end;

function TMyFileDialogEvents.OnFileOk(const pfd:
IFileDialog): HResult;
begin
  Result := E_NOTIMPL;
end;

function TMyFileDialogEvents.OnFolderChange(const
pfd: IFileDialog): HResult;
begin
  Result := E_NOTIMPL;
end;

function TMyFileDialogEvents.OnFolderChanging(
const pfd: IFileDialog;
  const psiFolder: IShellItem): HResult;
begin
  Result := E_NOTIMPL;
end;

function TMyFileDialogEvents.OnOverwrite(const
pfd: IFileDialog;
  const psi: IShellItem; out pResponse: DWORD):
HResult;
begin
  Result := E_NOTIMPL;
end;
```

```
function
TMyFileDialogEvents.OnSelectionChange(const pfd:
IFileDialog): HResult;
begin
  Result := E_NOTIMPL;
end;

function
TMyFileDialogEvents.OnShareViolation(const pfd:
IFileDialog;  const psi: IShellItem; out
pResponse: DWORD): HResult;
begin
  Result := E_NOTIMPL;
end;

function TMyFileDialogEvents.OnTypeChange(const
pfd: IFileDialog): HResult;
begin
  Result := E_NOTIMPL;
end;

function TMyFileDialogEvents.OnItemSelected(const
pfdc: IFileDialogCustomize; dwIDCtl: DWORD;
dwIDItem: DWORD): HResult;
begin
  Result := E_NOTIMPL;
end;
```

```
function
TMyFileDialogEvents.OnButtonClicked(const pfdc:
IFileDialogCustomize; dwIDCtl: DWORD): HResult;
var
  iImageEnIO: TImageEnIO;
  iFilename: string;
  iName: PChar;
begin
  if dwIDCtl = dwVisualGroup8ID then
  begin
    iImageEnIO := TImageEnIO.Create(nil);
    try
      with TFileSaveDialog(Form1.FileSaveDialog1)
      do
        Dialog.GetFileName(iName);
      iFilename := string(iName);
      if FileExists(FFilename) then
      begin
        iImageEnIO.LoadFromFile(FFilename);
        // Do IO Preview
        iImageEnIO.PreviewsParams :=
           [ioppDefaultLockPreview];
        iImageEnIO.SimplifiedParamsDialogs :=
           False;
        iegDefaultPreviewsZoomFilter := rfNone;
        iImageEnIO.DoPreviews([ppAuto]);
      end;
    finally
      iImageEnIO.Free;
    end;
    Result := S_OK;
  end
  else
  begin
```

```
    Result := E_NOTIMPL;
  end;
end;

function
TMyFileDialogEvents.OnCheckButtonToggled(const
pfdc: IFileDialogCustomize; dwIDCtl: DWORD;
bChecked: BOOL): HResult;
begin
  Result := E_NOTIMPL;
end;

function
TMyFileDialogEvents.OnControlActivating(const
pfdc: IFileDialogCustomize; dwIDCtl: DWORD):
HResult;
begin
  Result := E_NOTIMPL;
end;
```

Demo Applications

Although a book is good for study and tips, nothing beats demonstration applications. For this reason a number of demonstration programs were developed along with this book to show many of the tips and how to's referenced in this book. The demonstration applications were built with Delphi 2010. Versions for other Delphi compilers are not available, although I suspect many or all of them should compile with Delphi XE or XE2. The demonstration examples are included in the zip file containing this document.

Getting Started With ImageEN

The demonstration programs are:

1. ApprehendWithTIEImageListEx.zip- Shows how to use TIEImagelistEx which is a descendent of TIEImagelist.
2. IEWin7FileDialogs.zip- demonstrates used of a new FileDialogs descendent component to display file params in the Open and Save dialogs and how to add an “Advanced” button to the Save dialog.
3. ImageEnViewDrawing.zip- Shows how to draw with GDI and GDIPlus with ImageEn.
4. ImageEnBitDepthWithTIEImageListEx.zip- Shows how to change the bitdepth of images and save the result to a TIEImageListEx.
5. ImageEnImageList.zip- Shows how to use TIEImageList.
6. ImageEnResourceExtractor.zip- Shows how to view and extract image resources from dll’s
7. ImageEnViewBitDepth.zip- Show how to set and get the ImageEnView bitdepth.
8. ImageEnViewDrawBackBuffer.zip- Shows how to draw in the ImageEnView DrawBackbuffer event.
9. ImageEnViewLayersDragAndDrop.zip. Shows how to create layers with drag and drop. The demo can actually be used to modify glyphs for your own applications.
10. ImageProperties.zip- Shows how to display an ImageEn properties dialog to display image information, IPTC and EXIF information.
11. TFileDialogsWithImageEn.zip- Shows how to use the Vista and Windows 7 file dialogs with ImageEn to display file params in the dialogs and how to add an “Advanced” button to the TFileSaveDialog.

Chapter 2

Features

Chapter 2. Features

Input/Output

- Asynchronous load/save operations
- Load and save JPEG (RGB, GrayScale, YCbCr, CMYK, and YCbCrK) to file or stream (with also ½, ¼ and 1/8 subsizes for fast preview).

- Read/write APP0-APP15, COM Jpeg markers.
  JPEG2000: JP2, J2K and JPC code stream formats (JPEG-2000 Part-1 standard, ISO/IEC 15444-1)
- ReadOnly baseline support for single page and multipage DICOM (for medical imaging)
- Jpeg Lossless transformations and cut
  Estimation of Jpeg file quality
- Optional Color Management System (CMS) to process ICC profiles
- Support for Microsoft Color Management System to process ICC profiles

ImageEN
Features

- Read/write of Exif information (in Jpeg, TIFF, RAW, PSD and HD Photo). Can replace EXIF information without loading or changing the original image. Read/write EXIF maker note. Read/write EXIF-GPS tags
- Read/write Adobe XMP info from Jpeg, TIFF, HD Photo and PSD file formats.
- Load and save TIFF (rev.6.0 and Tech. Note #2, Packbits, JPEG, LZW, CCITT G.3 and G.4, ZIP) with RGB, CMYK, B/W, CIELab color spaces to file or stream. Also FAX (CCITT3), G3F and G3N (Zetafax) formats supported for loading. Supports paletted 4, 8, 16, 32, 64, 128 and 256 colors images. Supported Adobe Deflate compression (ReadOnly).
- Load and save Adobe PSD files. Handling of multiple layers.
- Load and save Microsoft HD Photo (also named WMP)
- Available external plugins to load and/or save JBIG, RAW, FITS, PCD and many others.
- Powerful class to handle TIFF files (add/edit/remove tags, add/delete pages, merge TIFFs, etc.)
- Supported several RAW Camera formats (CRW, NEF, ..)
- Native operations on TIFF files supports
- Add/remove/extract/enum without decompressing resulting in 450 times faster processing
- Read of single TIFF tags
- Save PostScript (PS and EPS) format - single and multipage using RLE, G3Fax, G4Fax and Jpeg compression
- Save Adobe PDF format - single and multipage images using G3Fax, G4Fax and Jpeg compression
- Load and save PNG with various compression levels
- Load and save compressed and uncompressed BMP in 2,16, 256 or 16M color images to files or streams
- Load and save uncompressed and black/white WBMP (Wireless Bitmap)
- Load and save compressed and uncompressed PCX images in 2,16, 256 or 16M colors to file or stream

ImageEN
Features

- Support for RAW files, specifying width, height, color format, bit alignment, etc.
- Load and save 2, 4, 8,16, 32, 64, 128 or 256 color GIFs to files or streams
- Load and save DCX (multipage PCX) files
- Load and save DIB, RLE, TGA (TARGA, VDA, ICB, VST, PIX)
  Load and save Portable Bitmap PBM, PGM and PPM
- Load and save ICO (provisions to compose ICO with multi resolution and color depth)
- Load WMF, EMF and CUR
  Load and save AVI, animated GIF and multi-page TIFF
- Load and save Window Media formats (MPEG, WMV, etc...), using DirectShow api
- Read of a single AVI frame
- Read/Write IPTC (IIMV4) information (like PhotoShop file info) from JPEG and TIFF.
- Can replace IPTC information without loading or changing original image
- Support for user defined file formats
  Support for Alpha Channel (transparent and semi-transparent images)
- Editing animated GIF: extract, insert and delete single images
- Editing multi TIFF: extract, insert and delete single images
  Included source code of a sample plug-in to read/write LZW Gif and TIFF
- Get image properties without loading
- Image acquisition from TWAIN scanners with full control of scanner capabilities and without default scanner user interface.
- Supports both modal and modeless acquisition
- Image acquisition from WIA scanners and cameras
  Save/load specific file format parameters and save/load preview dialogs (e.g. you can set quality of a Jpeg and immediately view the quality loss). Places bar on Win2000/Xp/Me.

ImageEN
Features

- A property to select language used in dialogs: English, Italian, German, Spanish, French, Portuguese, Greek, Russian, Dutch, Swedish, Polish, Japanese, Czech, Finnish, Farsi, Chinese (experimental), Danish, Turkish, Hungarian or customized by user.
- Supports connections with standard TImage component.
- OnProgress/OnFinish events for monitoring saving and loading.
- Support for aborting input/output processing
  Load alpha channel for GIF, TIFF, PNG, ICO, CUR, TGA, PSD
- Capture from screen
  Load images from the network, using http/ftp protocol (LoadFromURL)
- Registration of ImageEn formats in TPicture (TImage and VCL open/save dialogs)

- Wang Imaging annotations (load/edit/save)
- Read/write XP properties of jpeg and tiff files.
  Quality CMYK->RGB and RGB->CMYK conversion using an embedded profile
- RFB (VNC) client to handle a single or multiple VNC connections

Printing

- Printing (and print preview) functions to print single images specifying page alignment or absolute position
- Printing of multiple images (organize images in a sheet or in multiple sheets)

ImageEN
Features

Video Capture

- VCL visual component for video capture (freeze frames, save AVI files and real time frames processing).
- Supports all codec installed and video capture cards
  VCL non-visual component for video capture (capture without display the video input)
- Settable audio capture parameters
  Video capture: using DirectShow (WDM) capture drivers. This allows video sources to be selected by code from video input, tuner, etc.
- Video capture of multimedia files supported by DirectShow
  Saving of multimedia files with compressions supported by DirectShow

Image Processing & Analysis:

- Flexible image area selections (rectangle, ellipse, polygon and magicwand) including “soft” selection (you specify the selection intensity from 0 to 100%) and feathering selections
- Regulation of contrast, HSL, HSV/HSB and RGB components
- 11 filters for quality resampling (Triangle, Hermite, Bell, BSpline, Lanczos3, Mitchell, Nearest, Linear, FastLinear, Bilinear, Bicubic, ProjectBW, ProjectWB)
- Application of 3x3 filters
- Several automatic color/contrast/luminosity enhancement algorithms
- Merge two images
- Crop selected region
- Fast Fourier Transformation (FFT) with preview

ImageEN
Features

- Frequency domain (FFT) filtering
  Preview of all applied effects in a single dialog, with 8 preset filters (blur, edge detection, emboss, highpass1/ 2/ 3, lowpass1/ 2) and save and load filters from files
- Conversion to gray scale and negative
  Conversions of true color to black/white images with ordered or threshold dithering
- Casting of a group of colors to a single color
- Rotate to any angle (with enhanced quality)
  Horizontal and vertical flip
- Edge detection (to convert color images to black/white using edge detection algorithms)
- Skew detection (to estimate the orientation angle of the lines of text)
- Simple algorithm which helps to remove red eye effect
- Bump mapping effect
  Lens effect
- Wave effect
  Write/read hidden text, picture and raw data inside images (pixel amplitude modulation)
- Copy, cut, paste and paste into a rectangle of all or part of the image
- Multilevel Undo/Redo operations
- Application effects to the selected zones (also irregular)
- OnProgress event for monitoring lengthy image processing tasks
- Histogram equalization (auto and manual) and threshold
- Median cut and Neural network color quantizers, for fast and accurate color reduction
- Reduction to any number of colors (with colormap output)
- HSV channel separations
  A property to select language used by dialogs: English, Italian, German, Spanish, French, Portuguese, Greek, Russian, Swedish, Polish, Japanese, Czech, Finnish, Farsi, Chinese (experimental), Danish, Turkish, Hungarian or customized by user

ImageEN
Features

- A dialog to view a palette and select a color from it
  VCL component for HSV/HSB color selection
- VCL component for displaying histogram of gray levels (or RGB channels)
- Connections with standard TImage component
- Maximum (dilation), Minimum (erosion), Opening and Closing filters with preview
- Horizontal/vertical pixel density histograms
- Noise removal from Black/white images
- Gamma correction
- Gaussian Blur effect
- Soft Shadow effect
- Inner Shadow effect
  Sharpening effect
- Motion blur effect
  Optional Color Management System (CMS) to process ICC profiles
- Manual correction of Barrel Distortion and Pincushion distortion (lens distortion, underwater distortion)
- Supported following native pixel formats: 1bit, 8 bit paletted, 8 bit gray scale, 16 bit gray scale, 24 bit RGB, 32 bit float point, 24 bit CMYK, 48 bit RGB, CIELab
- Crop and AutoCrop functions
- Encrypt/decrypt function using a 128 bit symmetric key algorithm
- Adjust of color temperature

ImageEN
Features

Image Rendering

- Real time Zoom-in and zoom-out (by percentage, zoom-in rectangular area or by one mouse click, accepts floating point values)
- Support for multiple layers with separated transparency and layer mask
- Select from 37 layer operations (blend modes)
- Moveable, resizeable and rotatable layers (by code or by user interaction)
- Magnify layers (rectangle or like a lens/glass)
- More than 200 transition effects
- Quality zoom effects with six filters (select between fast zoom or quality zoom)
- Alpha Channel (transparent images)
- B/W filtered zoom-in and zoom-out (for optimal display large B/W images inside small rectangles)
- Scrollbar in two dimensions (real time) for the fast exploration of images, and “hand navigation” (click and drag the image with mouse)
- Animate bi-color contour of a selected region (polygonal, irregular, circular and rectangular selections)
- Magic wand selection (inclusive, exclusive and global)
- Multiple selections
- Sizeable and moveable selected regions
- Save/Restore of selections
- Multi-polygonal animated regions
- Dithering for non-true color display adapters
- VCL component to implement interactive rules with grips and numeric labels
- VCL component for interactive gradient bar
- Full support for mouse wheel actions

- Quality view for projects (ZoomFilter = ProjectBW and ProjectWB)

Thumbnails

- Coverflow-like effects to present images
- Single VCL component to display and animating thumbnails and grids of images.
- Handles big image sequences without allocating system memory
- 3D style and customized thumbnails
- Save a snapshot (save images, caches, texts...) in a single file to instantly reload the component content
- Multi-selection
- Load/Save AVI, GIF, DCX, multimedia formats (using DirectShow), multi-page TIFF images
- Allows to set a wallpaper (background image)
- Multi-threading image loading
- Cache to speed-up image rendering
- Show a custom or preset background under thumbnails
- Loading of EXIF thumbnails to speedup display
- Load of entire directories of images using multiple threads and extracting EXIF thumbnails

ImageEN
Features

Database Handling

- Data aware component (TImageEnDBView) integrated with Delphi database environment (saves and loads Bmp, Pcx, Jpeg, TIFF, PNG, TGA, PBM, PGM, PPM and Gif inside Blob fields or path reference).
- Data aware component (TImageEnDBVect) integrated with Delphi database environment (saves and loads Bmp, Pcx, Jpeg, TIFF, PNG, TGA, PBM, PGM, PPM and Gif and vectorial objects inside Blob fields or path reference).

ImageEN
Features

Vectorial Objects Handling

- Vectorial objects (lines, box, circles, ellipses, bitmaps, text, multiline text, static ruler, polylines, polygons, angles, arrows-with text, curved text, highlight box) over background image.
- Save/Load these objects and import a sub-set of Autocad DXF files.
- Undo/Redo capability.
- Support to add/edit/delete points of a polygon.
- Anti-alias capability.
- Settable object transparency.
- Object soft shadow effects.
- Measurement of lines, perimeters and areas (in pixel, inch, cm, mm or Km units).
- Angle measurements.
- Saves/loads IEV format (compressed with Deflate, variation of LZ77 algorithm) that includes images and vectorial objects.
- Anti-aliased text.
- Alpha channel for images objects (transparent images).
- Clipboard Cut/Copy/Paste of vectorial objects.
- Create polygons from raster image edges.
- Remove jagged edges from polygons.
- Polygon simplification.

ImageEN
Display

TImageEnView is the primary ImageEn component to display and edit images. TImageEnVect is similar to ImageEnView except it adds support for adding vectorial objects such as lines, boxes, ellipses and text on top of the background image. TImageEnMView is the same as TImageEnView except it adds support for handling multiple images.

TImageEnView
TImageEnMView
TImageEnVect

Chapter 3

TImageEnView

Chapter 3. ImageEnView

TImageEnView is an image container and viewer. TImageEnView allows zooming, scrolling and can handle multiple layers. TImageEnView encapsulates a TImageEnIO (IO property) and TImageEnProc (Proc property) components, to allow input/output and image processing tasks without adding TImageEnIO and TImageEnProc components to the projects.

Usually; however, ImageEnIO and ImageEnProc should be placed in uses.

TImageEnView is the primary ImageEn component that can be used to display and edit images.

Unit ImageEnView
TImageEnView

Although TImageEnIO is listed separately as a stand-alone component in this document as well as in the ImageEn Help file the IO functions are accessible with TImageEnView by using ImageEnView1.IO methods. So it is not necessary to attach a TImageEnView to a TImageEnIO component to the access input/output functions in TImageEnIO.

Similarly, the TImageEnProc methods are accessible with TImageEnView by using ImageEnView1.Proc methods so it is not necessary to attach a TImageEnView to a TImageEnProc component to the access image processing functions contained in in TImageEnProc.

ImageEn provides many enhancements that is difficult to do with TImage. The primary advantages are access to TBitmap and TIEBitmap, Selections, Layers, Frames, and FileIO for most of the popular file types including PNG Image and TIF Image, and Jpeg and many image processing functions.

ImageEn has a large advantage over TImage by providing a TIEBitmap over TBitmap. TIEBitmap is a replacement of VCL TBitmap class. It has many methods and properties compatible with TBitmap and enhances it by supporting multi-threading, large images, and access to the alphachannel. One of the great features of ImageEn is that if you make any changes to TIEBitmap via the ImageEnView1.IEBitmap this change is reflected by the TBitmap. Thus working with ImageEn with 32-bit images with alphachannel that is popular today is made simple. The ImageEnView1.Bitmap is readily used to provide access to the image with other third party components and classes.

Copying images, creating previews, handling layers and multi-frame images could not be easier.

Unit ImageEnView
TImageEnView

Copying Images From One ImageEnView To Another

Images can be copied from one ImageEnView to another ImageEnView by using the Assign method:

```
ImageEnView1.Assign( ImageEnView2 );
```

or by assigning the bitmap or IEBitmap as follows:

```
ImageEnView1.Assign( ImageEnView1.Bitmap );
// this doesn't copy DPI, but just the image
```

Thumbnail Previews

Thumbnail previews can be created in a number different ways. One way is to assign ImageEnView1 to ImageEnView2 and setting the ImageEnView AutoShrink property to true. Another way is to use the SetExternalBitmap method that makes it possible to connect a TImageEnView component to another one, sharing the same bitmap. This is useful to view the same image with multiple TImageEnView components, loading the image only one time.

```
ImageEnView2.SetExternalBitmap(
ImageEnView1.IEBitmap );
```

A disadvantage of SetExternalBitmap is that you may not do anything to the bitmap in ImageEnView2 when SetExternalBitmap is active. If you do, you are likely to get an exception. You may “unhook” the external bitmap by setting SetExternalBitmap to nil however.

```
ImageEnView2.SetExternalBitmap(nil);
```

I have found… at least for me it is preferable to just handle your own preview thumbnail in your own code which leaves you in full control of the preview, especially if you are using layers and you wish to see the layers in the preview.

Handling Layers In A Preview

Handling layers in a preview is a little more complicated but can easily be accomplished by saving the ImageEnView1 layers to a TMemory stream then by loading the layers into ImageEnView2 from the stream in the ImageEnViewLayerNotify event as follows:

Unit ImageEnView

TImageEnView

```
procedure TForm1.ImageEnViewLayerNotify (Sender:
TObject; layer: integer; event: TIELayerEvent);
var
  iMS: TMemoryStream;
begin
  if (layer <> 0) and (event = ielSelected) then
    begin
      if (event = ielMoving) or (event = ielResized)
        then
          begin
            iMS := TMemoryStream.Create;
            try
              ImageEnView1.LayersSaveToStream (iMS);
              // Important
              iMS.Position := 0;
              ImageEnView2.LayersLoadFromStream
               (iMS);
              ImageEnView2.Update;
            finally
               iMS.Free;
            end;
          end;
    end;
end;
```

If you handle your preview in this manner all layers will be visible in the preview and the preview will be updated when you move or resize a layer.

See the ImageEnViewLayersDragAndDrop Demo in the zip file containing this document for a demo depicting handling layers in a preview TImageEnView.

Unit ImageEnView

TImageEnView

Getting Image Properties

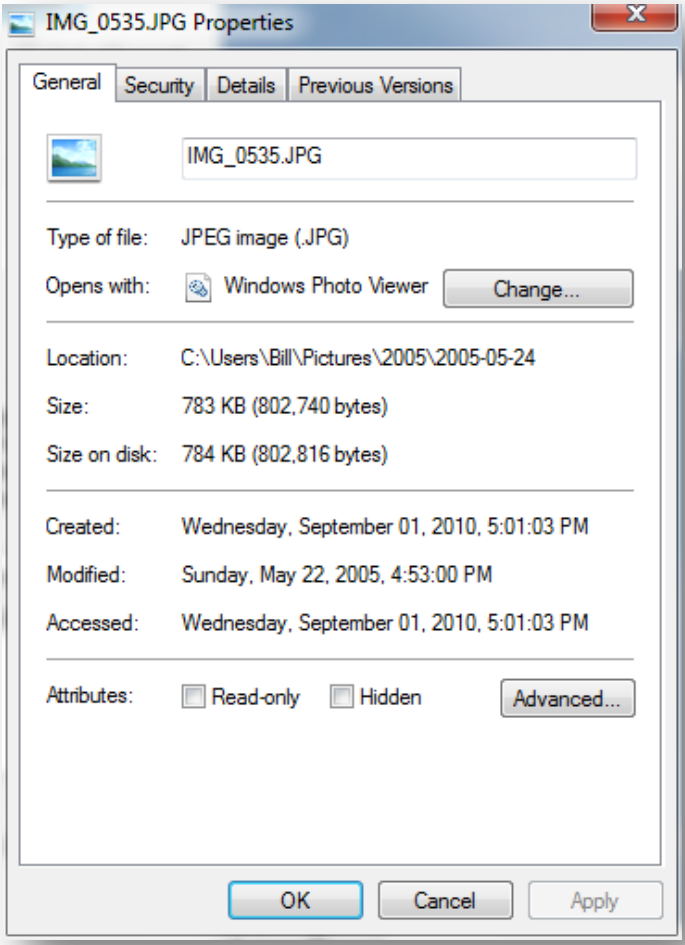
The simplest way to display a image properties dialog is to call ShowProperties(Application.Handle, ImageEnView1.IO.Params. FileName) as shown below:

```
function ShowProperties
  (hWndOwner: HWND; const FileName: string)
  : boolean;
var
  Info: TShellExecuteInfo;
begin
  { Fill in the SHELLEXECUTEINFO structure }
  with Info do
  begin
    cbSize := SizeOf(Info);
    fMask := SEE_MASK_NOCLOSEPROCESS or
      SEE_MASK_INVOKEIDLIST or
      SEE_MASK_FLAG_NO_UI;
    wnd := hWndOwner;
    lpVerb := 'properties';
    lpFile := pChar(FileName);
    lpParameters := nil;
    lpDirectory := nil;
    nShow := 0;
    hInstApp := 0;
    lpIDList := nil;
  end;
  { Call Windows to display the properties
dialog. }
  Result := ShellExecuteEx(@Info);
end;
```

Unit ImageEnView

TImageEnView

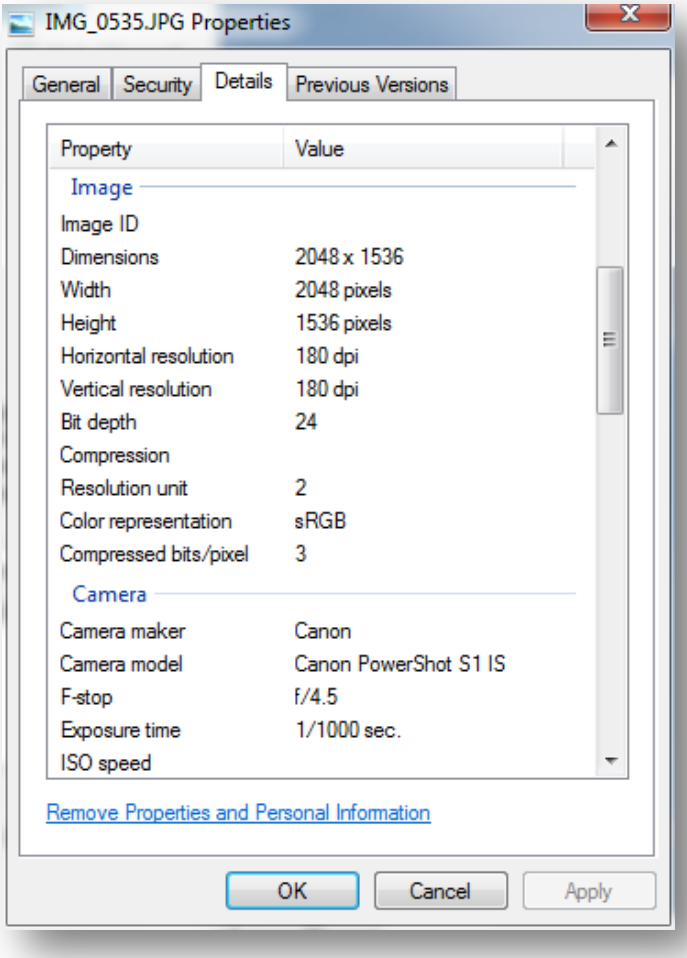
The Window 7 Native properties Dialog- General Tab is depicted below:



Unit ImageEnView

TImageEnView

Window 7 Native properties Dialog-Details Tab is depicted below:



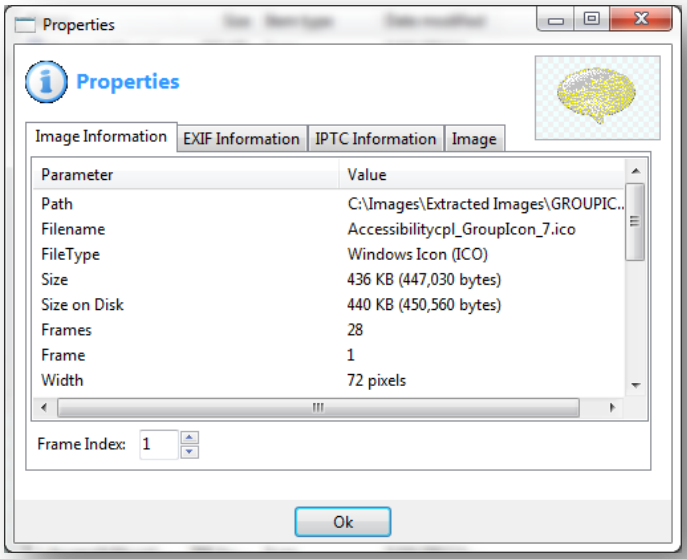
Unit ImageEnView

TImageEnView

The native windows 7 properties dialog includes some image information along with some EXIF information is demonstrated in the ApprehendWithTIEImageListEx.zip file in the zip file included with this document,

A more elaborate properties dialog that uses ImageEnIO to display image information, EXIF information, IPTC information, an image preview with multi-frame image support is demonstrated in the ApprehendWithTIEImageListEx.zip file accompanying this document. This dialog uses about 2,500 lines to display image properties information along it provides more capability that the system properties dialog provided by windows, especially for multi-frame images like icons. The zip file contains all of the source code.

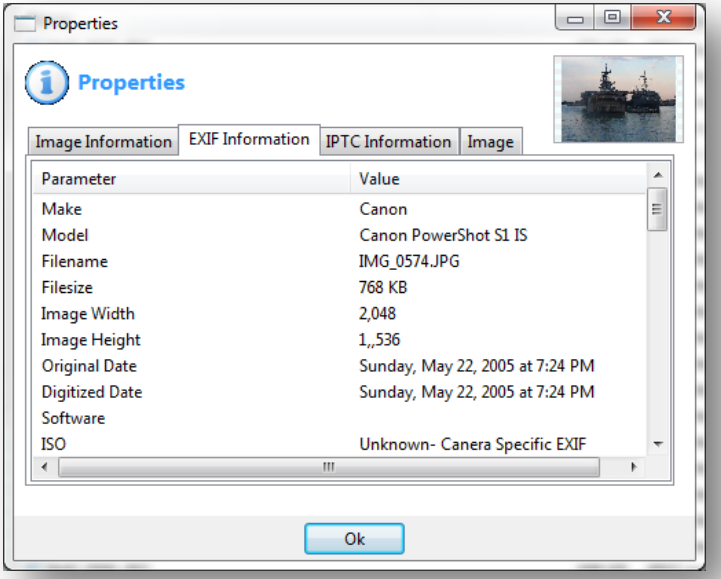
ImageENIO Properties Dialog- Image Information Tab is depicted below:



Unit ImageEnView

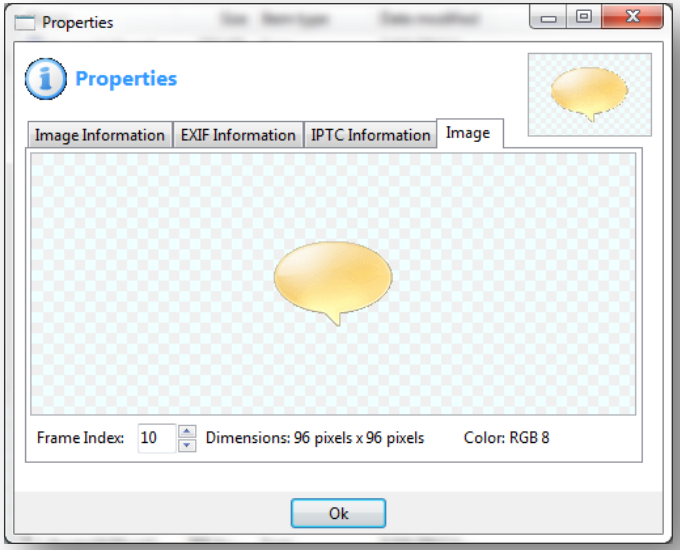
TImageEnView

ImageENIO Properties Dialog- EXIF Information Tab is depicted below:



Unit ImageEnView
TImageEnView

ImageENIO Properties Dialog- Image/Frame Tab is depicted below:



Unit ImageEnView
TImageEnView

Using Vista or Windows 7 Picture Dialogs

Standard Delphi OpenPictureDialog and SavePictureDialogs can be easily used with ImageEnView rather than by using the TimageEnOpenDialog and the TImageEnSaveDialog. The main reason for doing this to use the latest file dialogs available for your operating system. A disadvantage is you have to handle the file types you want to support by setting the SavePictureDialog's filter as well as handle changing the SavePictureDialog's file type to automatically update the filename extension by way of your own code.

Modifying The Picture Dialog's Filter

OpenPictureDialog Filter

```
procedure TForm1.Open1Click(Sender: TObject);
// Build the OpenPictureDialog filter
begin
  OpenPictureDialog1.Filter := '';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
    'All Supported File
Types|*.tif;*.gif;*.jpg;*.pcx;*.bmp;*.ico;*.cur;*.png;*.wmf
;*.emf;*.tga;*.pxm;*.jp2;*.j2k;*.avi;*.wbmp;*.ps;*.pdf;*.dc
x;*.raw;*.bmpraw;*.wmv;*.mpg;*.psd;*.iev;*.lyr;*.hdp;*.wdp|
';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
    'TIFF Bitmap (TIF)|*.tif|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
    'CompuServe Bitmap (GIF)|*.gif|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
    'JPeg Bitmap (JPG)|*.jpg|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
    'PaintBrush (PCX)|*.pcx|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
    'Windows Bitmap (BMP)|*.bmp|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
    'Windows Icon (ICO)|*.ico|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
    'Windows Cursor (CUR)|*.cur|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
    'Portable Network Graphics (PNG)|*.png|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
    'Windows Metafile (WMF)|*.wmf|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Windows Enhanced Metafile (EMF)|*.emf|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Targa Bitmap (TGA)|*.tga|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Portable Pixmap, GreyMap, BitMap (PXM)|*.pxm|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Jpeg2000 (JP2)|*.jp2|';
```

```
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Jpeg2000 (J2K)|*.j2k|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'AVI Video (AVI)|*.avi|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Wireless Bitmap (WBMP)|*.wbmp|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Postscript (PS)|*.ps|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
  'Adobe PDF (PDF)|*.pdf|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Multipage PCX (DCX)|*.dcx|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
  'Digital Camera RAW (RAW)|*.raw|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Bitmap RAW (RAW)|*.bmpraw|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Windows Media (WMV)|*.wmv|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Video MPEG (MPEG)|*.mpg|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Adobe Photoshop PSD (PSD)|*.psd|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'IE Vectorial objects (IEV)|*.iev|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'IE Layers (LYR)|*.lyr|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
  'Microsoft HD Photo (HDP)|*.hdp|';
  OpenPictureDialog1.Filter := OpenPictureDialog1.Filter +
   'Microsoft Media Photo (WDP)|*.wdp|';
  if OpenPictureDialog1.Execute then
  begin
    Caption :=   OpenPictureDialog1.FileName;
    ImageEnView1.IO.LoadFromFile(
      OpenPictureDialog1.FileName );
   …
end;
```

Unit ImageEnView
TImageEnView

SavePictureDialog Filter

```
procedure TMain1.SaveAs1Click(Sender: TObject);
var
  i: integer;
  iFileType: TIOFileType;
  iFrames: array of TObject;
  iFilePath: string;
  iExtension: string;
begin
  iExtension := Lowercase (ExtractFileExt(
    ImageEnView1.IO.Params.FileName));
  // Build the save picture dialog filter
  SavePictureDialog1.Filter := '';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'JPEG Bitmap (*.jpg; *.jpeg)|*.jpg;*.jpeg|';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'TIF Bitmap (*.tif; *.tiff)|*.tif;*.tiff|';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'Compuserve Bitmap (*.gif)|*.gif|';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'PaintBrush (*.pcx)|*.pcx|';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'Windows Bitmap (*.bmp; *.dib)|*.bmp;*.dib|';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'Windows Icon (*.ico)|*.ico|';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'Portable Network Graphics (*.png)|*png|';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'Targa Bitmap (*.tga)|*.tga|';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'Portable Pixmap (*.pxm)|*.pxm|';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'Wireless Bitmap (*.wbmp)|*.wbmp|';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'JPeg 2000 (*.jp2)|*.jp2|';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'JPeg2k (*.j2k)|*.j2k|';
  SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'PostScript Level 2 (*.ps;*.eps)|*.ps;*.eps|';
```

Unit ImageEnView

TImageEnView

```
SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'Adobe PDF (*.pdf)|*.pdf|';
SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'Multipage PCX (*.dcx)|*.dcx|';
SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'Photoshop PSD (*.psd)|*.psd|';
SavePictureDialog1.Filter := SavePictureDialog1.Filter +
   'Microsoft HD Photo (*.hdp;*.wdp) |*.hdp;*.wdp|';
if (iExtension <> '.wmf') and (iExtension <> .emf') then
  begin
    SavePictureDialog1.FileName := JustName(
      ImageEnView1.IO.Params.FileName) + iExtension;
    SavePictureDialog1.FilterIndex :=
      IEFileTypeToGraphicFilterIndex(
        ImageEnView1.IO.Params.FileType);
  end
  else
  begin
    SavePictureDialog1.FileName := JustName(
      ImageEnView1.IO.Params.FileName) + '.png';
    SavePictureDialog1.FilterIndex := 7;
  end;
  SavePictureDialog1.DefaultExt := '.png';
  if SavePictureDialog1.Execute then
  begin
    Screen.Cursor := crHourglass;
    try
      iFilePath := SavePictureDialog1.FileName;
      iExtension := ExtractFileExt (iFilePath);
      iFileType := IEExtToFileFormat (iExtension);
      if (iFileType = ioICO) or (iFileType = ioGIF
        ) or (iFileType = ioTIFF) then // Icon
      begin
        // Set the frames from the IEImageList
        SetLength (iFrames,FIEImageList.ImageCount);
        for i := 0 to FIEImageList.ImageCount - 1 do
        begin
          ImageEnView1.IEBitmap.Assign(
            FIEImageList.Image[i]);
          iFrames [i] := ImageEnView1;
          ImageEnView1.IO.Params.ICO_ImageIndex := i;
```

```
        end;
        // Save the icon
        IEWriteICOImages(iFilePath, iFrames);
        // Reload the first frame
        ImageEnView1.IEBitmap.Assign(
          FIEImageList.Image[0]);
        ImageEnView1.Update;
        ImageEnView1.Proc.ClearAllUndo;
        ImageEnView1.Proc.ClearAllRedo;
        Undo1.Enabled := ImageEnView1.Proc.UndoCount
          >0;
        Redo1.Enabled := ImageEnView1.Proc.RedoCount
          > 0;
      end
      else
      begin
        ImageEnView1.IO.Params.FileType := iFileType;
        ImageEnView1.IO.SimplifiedParamsDialogs :=
          False;
        ImageEnView1.IO.PreviewsParams := [
          ioppDefaultLockPreview];
       iegDefaultPreviewsZoomFilter := rfNone;
        if ImageEnView1.IO.DoPreviews([ppAUTO]) then
          ImageEnView1.IO.SaveToFile(iFilePath);
      end;
      Caption := iFilePath;
    finally
      Screen.Cursor := crDefault;
    end;
  end;
end;
```

Unit ImageEnView

TImageEnView

```
function IEFileTypeToGraphicFilterIndex (const
AIEFileType: TIOFileType): integer;
// Returns the Graphic Filter Index from the
// AIEFileType as an integer
begin
  case AIEFileType of
    ioJPEG: Result := 1;
    ioTIFF: Result := 2;
    ioGIF: Result := 3;
    ioPCX: Result := 4;
    ioBMP: Result := 5;
    ioICO: Result := 6;
    ioPNG: Result := 7;
    ioTGA: Result := 8;
    ioPXM: Result := 9;
    ioWBMP: Result := 10;
    ioJP2: Result := 11;
    ioJ2K: Result := 12;
    ioPS: Result := 13;
    ioPDF: Result := 14;
    ioDCX: Result := 15;
    ioPSD: Result := 16;
    ioHDP: Result := 17;
  else
    Result := -1;
  end;
end;
```

Unit ImageEnView
TImageEnView

Automatic Update Of The SavePictureDialog's File Extension

The SavePictureDialog's file extension will be automatically change to the selected file type filter by handling the SavePictureDialog's Type Change Event.

```
procedure TForm1.SavePictureDialog1TypeChange (
Sender: TObject);
var
  FilePath: string;
  FileName: string;
  FileExt: string;
begin
  FilePath := SavePictureDialog1.FileName;
  FileExt := ExtractFileExt(FilePath);
  FileName := ExtractFileName(FilePath);
end;
```

Loading Multi-Frame Files

The key to loading multi-frame file formats is setting the index to the frame to load as well as storing each frame for display or editing. Frames may be stored in memory with the IE class or in components or loaded at runtime in a single TImageEnView by setting the index of the frame to be loaded before loading the file.

A popular way to display multi-frame files is to create a TImageEnView on a TPageControl Tab sheet at runtime. When using ImageEn in this manner the properties and events for each TImageEnView must be set as well. Typically this is accomplished by determining the number of frames in the file then storing a description of the frame in a TListView and filling a TPageControl with child TImageEnView's with the images of each frame.

Unit ImageEnView

TImageEnView

Accessing Each Frame

Each frame of a multi-frame image may be displayed with TImageEnView as follows:

```
procedure TForm1.OpenFile(iFilename: string);
var
  iFileType: TIOFileType;
  iFrameIndex: integer;
begin
  if FileExists(iFilename) then
  begin
    Screen.Cursor := crHourglass;
    try
      ImageEnView1.IO.ParamsFromFile(iFilename);
      iFileType := IEExtToFileFormat(
        ExtractFileExt(iFilename));
      // Get the frame index to display
      iFrameIndex := StrToInt(
        EditFrameIndex1.Text);
```

See the ImageEnViewLayersDragAndDrop Demo in the zip file containing this document for a demo depicting the use of Windows 7 Picture Dialogs with ImageEn..

```
      case iFileType of
        ioICO:
          ImageEnView1.IO.Params.ICO_ImageIndex
            := iFrameIndex;
        ioGIF:
          ImageEnView1.IO.Params.GIF_ImageIndex
            := iFrameIndex;
        ioTIFF:
          ImageEnView1.IO.Params.TIFF_ImageIndex
            := iFrameIndex;
      end;
      // Load the frame based in the index
      ImageEnView1.IO.LoadFromFile(iFilename);
    finally
      Screen.Cursor := crDefault;
    end;
  end;
```

Loading All The Frames

TImageEnMView can load each frame in the image for display in a TImageEnView or the frames may be stored in memory with a TIEImageList. TIEImageList is a simple in memory list of TIEBitmap images.

Use a TIEImageList To Store Frames and TListView to store frame descriptions and frame indexes.

Important methods are shown in **blue.** This example uses a TListView, and a TImageEnView component and ImageEnIO created at runtime to load multi-frame images into a TIEImageList and a description of the frames in TListView.

Unit ImageEnView

TImageEnView

Open a multi-frame or single frame ICO, GIF or TIF file

Create a TIEImageList to hold the images in memory in OnFormCreate:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  FIEImageList := TIEImageList.Create;
  ImageEnView1.Proc.AutoUndo := False;
  ImageEnView1.Proc.UndoLimit := 99;
  ImageEnView1.SetChessboardStyle(8);
end;
```

Free the TIEImageList in FormDestroy:

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
  // Free the multi-frame imagelist
    FIEImageList.Clear;
    FIEImageList.Free;
end;
```

Unit ImageEnView

TImageEnView

Load a multi-frame or single frame image

```
procedure TForm1.Open1Click(Sender: TObject);
var
  i: integer;
  iFrames: integer;
  iFileName: string;
  iExtension: string;
  iImageEnIO: TImageEnIO;
  iBitDepth: integer;
  iColor: string;
  iListItem: TListItem;
begin
  if OpenDialog1.Execute then
  begin
    // Close all frames
    FIEImageList.Clear;
    ListView1.Clear;
    iFileName := OpenDialog1.FileName;
    iExtension := Lowercase(ExtractFileExt
                  (iFileName));
    iFrames := IEGetFileFramesCount(iFileName);
    // Load multi-frame image
    if iFrames > 1 then
    begin
      ListView1.Items.BeginUpdate;
      try
        for i := 0 to iFrames - 1 do
        begin
          iImageEnIO := TImageEnIO.Create(nil);
          try
            // Important- ImageIndex determines
            // the frame that will be loaded
```

```
if iExtension = '.ico' then
begin
  iImageEnIO.Params.ICO_ImageIndex := i;
  ImageEnView1.IO.Params.ICO_Background :=
     TColor2TRGB(clBlack);
end;
if iExtension = '.gif' then
   iImageEnIO.Params.GIF_ImageIndex := i;
if iExtension = '.tif' then
  iImageEnIO.Params.TIFF_ImageIndex := i;
iImageEnIO.LoadFromFile (iFileName);
// Add the image of the frame to IEImageList
FIEImageList.AppendImageRef
  (TIEBitmap.Create
  (iImageEnIO.IEBitmap), iFileName);
// Show each frames information in a TListView
iListItem := ListView1.Items.Add;
iListItem.Caption :=
  IntToStr(ListView1.Items.Count);
iListItem.ImageIndex := 10;
iListItem.SubItems.Add(iFileName);
iListItem.SubItems.Add(IntegerToString(
   ImageEnView1.IEBitmap.Width) + ' x ' +
   IntegerToString(ImageEnView1.IEBitmap.Height)
   + ' pixels');
iBitDepth :=
  ImageEnIO.Params.SamplesPerPixel *
     iImageEnIO.Params.BitsPerSample;
```

```
if iBitDepth = 32 then
  iColor := 'RGBA 32-Bit'
else
  iColor := 'RGB ' + IntToStr(iBitDepth)+ '-bit';
iListItem.SubItems.Add(iColor);
ListView1.ItemIndex := 0;
finally
  iImageEnIO.Free
end;
end;
finally
  ListView1.Items.EndUpdate;
end;
end
```

```
else // Load Single-frame file
begin
  if iExtension = '.ico' then begin
    ImageEnView1.IO.Params.ICO_ImageIndex := 0;
    ImageEnView1.IO.Params.ICO_Background :=
       TColor2TRGB(clBlack);
  end;
  if iExtension = '.gif' then
    ImageEnView1.IO.Params.GIF_ImageIndex := 0;
  if iExtension = '.png' then
    ImageEnView1.IO.Params.PNG_Background :=
       TColor2TRGB(clBlack);
  ImageEnView1.IO.LoadFromFile(iFileName);
  if ImageEnView1.IO.Aborting then
  begin
    MessageBeep (MB_ICONERROR);
    TaskDialog1.MainIcon := tdiError;
    TaskDialog1.Title := 'Error';
    TaskDialog1.Caption := 'Image Loading
                           Failed';
    TaskDialog1.Text := 'The image could not be
        loaded.';
    TaskDialog1.ExpandedText := 'The image could
       not be loaded and may be corrupt.';
TaskDialog1.CommonButtons := [tcbOk];
TaskDialog1.Execute;
exit;
end;
//ImageEnView1 displays a frame depending on
// the index set by the TListView's ItemIndex.
ImageEnView1.EnableAlphaChannel := True;
ImageEnView1.IO.Params.
```

Unit ImageEnView

TImageEnView

```
 BMP_HandleTransparency := True;
 iBitDepth :=
   ImageEnView1.IO.Params.SamplesPerPixel *
      ImageEnView1.IO.Params.BitsPerSample;
if iBitDepth = 32 then
   iColor := 'RGBA 32-bit'
else
   iColor := 'RGB ' + IntToStr(iBitDepth) + '-
             bit';
// Show each frames information in a TListView
iListItem := ListView1.Items.Add;
ListView1.Columns[0].Width := 50;
iListItem.Caption :=
  IntToStr(ListView1.Items.Count);
iListItem.SubItems.Add(
ImageEnView1.IO.Params.FileName);
iListItem.SubItems.Add (IntegerToString(
  ImageEnView1.Bitmap.Width) + ' pixels x ' +
IntegerToString(ImageEnView1.Bitmap.Height )
          + ' pixels' );
iListItem.SubItems.Add(iColor);
iListItem.ImageIndex := 1;
end;
end;
end;
```

Unit ImageEnView
TImageEnView

Get the index of the selected image from the Listview and load the frame stored in FIEImageList into ImageEnView:

```
procedure TForm1.ListView1SelectItem (Sender:
TObject; Item: TListItem; Selected: Boolean);
begin
  if Selected then
  begin
    // load the selected from memory image list
    ImageEnView1.IEBitmap.Assign(
      FIEImageList.Image[Item.Index]);
    ImageEnView1.Update;
    ImageEnView1.IEBitmap.UpdateFromTBitmap;
    // Show the filename of the frame in the
   // forms caption
    Caption:= FIEImageList.Filename[Item.Index];
  end;
end;
```

See the ImageEnImageList Demo in the zip file containing this document. This demo adds all opened and captured images to a memory TIEImageList and demonstrates ImageENIO Screen Capture, Open Picture dialogs, Save Picture Dialog with automatic filename change when file type changes, Cropping, Undo and Transitions.

Using Transitions

ImageEn transitions allow the developer to control how images appear when they are painted in ImageEn. Transitions are similar to the screen wipes and fades you see on television.

PrepareTransition copies the currently displayed image to an internal buffer. This allows you to change current image (load, update, etc.) before starting the transition with RunTransition. RunTransition starts the transition using Effect and Duration parameters. Effect specifies the effect. Duration specifies the duration of the transition in milliseconds.

An image must be present in TImageEnView before calling RunTransition. The existing image may be loaded from a file, from a stream or assigned from an image source. See the example code here.

TransitionTiming allows you to select how the transition progresses. There are three forms of transition timing. The default is iettLinear. IettLinear translation timing starts and ends the transition with the same speed. IettExponential translation timing starts the transition slowly and ends the transition moderately fast. IettLogarithmic translation timing starts the translation very fast and ends the translation very slowly.

TransitionRunning is true whenever a transition is running. TransitionTiming allows you to select how the transition progresses. AbortTransition aborts current transition started with RunTransition.

See the ImageEnImageList Demo in the zip file containing this document that demonstrates how to use Transitions.

ROI (regions of interest): ImageEn Selections

TImageEnProc provides image processing and analysis tasks on attached TImageEnView, TImageEnDBView, TImageEn, TIEBitmap, TImage or TBitmap components. Also, it handles all clipboard operations.

Unit ImageEnView

TImageEnView

It includes an area selection capability for most of image processing tasks (only if the attached component isn't TImageEnView, which already has the area selection capability).

Unit ImageEnView
TImageEnView

Using Vista or Windows 7 TOpenFile and TSaveFile Dialogs

Delphi 2009 introduced a new TFileOpenDialog and the TFileSaveDialog component for Vista. The new class implements the IFileDialog interface, so it ONLY works with Vista or above.

To customize Vista's and Windows 7 common file dialog, MS introduced the IFileDialogCustomize interface. Previous to Vista, the customization was done by a custom dialog template resource. This is how the IEOpenDialog and IESaveDialog components were built.

The IFileDialogCustomize can be queried from the TFileOpenDialog.Dialog or TFileSaveDialog.Dialog property. To get the IFileDialogCustomize interface, we some coding inside the TFileOpenDialog.OnExecute() or TFileSaveDialog.OnExecute()events.

The old customization method will not work under Vista unless the file dialog is restricted to XP's style. There is no way to make the old template method work with the Vista's native dialog. The only way is to make use of the IFileDialogCustomize interface. The major drawback of the new method is the extraordinary limitation/restriction in the types of controls at can be used with the dialogs. Firstly, the IFileDialogCustomize interface only works with Vista or above. Second, the choice of controls are very limited and **no** user-defined controls are allowed. The provided controls are restricted to Menu, Button. Combo box, Radio button list, Check button (check box), Edit, Separator and Label.

TImageEnView cannot be used for the preview so we are stuck with the dialogs built-in preview, but we can add labels to display image information as well as add an "Advanced"

button to call TImageENIO.DoPreviews. The added controls are pretty useless unless you can interact with them. There are IFileDialogEvents and IFileDialogControlEvents interfaces to deal with the controls. To implement it, simply declare a class:

TMyFileDialogEvents = class(TInterfacedObject, IFileDialogEvents, IFileDialogControlEvents)

On my Windows 7 machine, native preview support is provided for most of the popular image formats including Bitmaps **BMP**), Portable Network Graphics (**PNG**), Compuserve Bitmap (**GIF**), Jpeg Bitmap (**JPG**), TIF Bitmap (**TIF**) and The Windows Media Photo (**WDP**) file formats. The Windows Media Photo (WMP) format has transitioned to the Windows HD Photo format, which uses the .HDP file extension. The RAW File or DICOM formats were **Not** tested. The JPeg2000(**JP2**), JPeg2000 Code Stream (**J2K**), Portable Pixmap, GrayMap, BitMap (**PXM**;PPM;PGM;PBM), Windows HD Photo (**HDP**), Targa (**TGA**) file types are not supported by the native Windows preview on my pc.

The final step is to add ImageEn's file types support for the Vista/Windows 7 File Dialog's preview, but unfortunately a dll must be built using ImageEnIO file support, registered and installed to provide the additional file support. I decided not to pursue this because all of the file types required by me are supported by the native preview and because of having to write and install the dll.

The most important advantage of using the Windows Vista and higher file dialogs is the display of image information for the selected file in the dialog just like the ImageEn dialogs do.

Unit ImageEnView
TImageEnView

In the zip file accompanying this document there is a demo application that shows how to use the file dialogs with ImageEn. There is also a component along with its accompanying demo that implements a TIEFileDialogs which offers both Open and Save dialogs without having to write any code. Because of the limitations listed previously, only native windows preview support is provided by the component.

Unit ImageEnView
TImageEnView

TIEImageListEx

TIEImageList is one of my favorite ImageEn classes because of its versatility , ease of use and simple coding requirements. TIEImageList could easily be over looked by developers because it is a recent addition to ImageEn and because it is not a component. TIEImageList is an excellent tool to use especially with small images like glyphs, small bitmaps or icons. It can often replace a much harder to use TPageControl that contains a TImageEnView on each page which requires much more code to create the tabsheets and ImageEnView at runtime. So for a glyph or icon viewer or editor type of applications or picture viewer or editor it will be a valuable addition to your VCL toolkit.

TIEImageList is a class and is not a component. TIEImageList is a simple list of TIEBitmap images. TIEImageList contains just pointers to TIEBitmap objects but it is very similar to TImageList, except it is non-visual and can hold any image supported by ImageEnIO of any dimensions or color depth.

The TIEImageList class can store TIEBitmaps in either memory and/or a file. The storage location is set by the TIEBitmap.Location property. Storage locations can be different for any image. The TIEImageList automatically sets the storage location if the image cannot be stored in memory. For example, if you set TIEBitmap.Location := ieMemory, but the actual image cannot be stored in memory, then Location for that image is automatically set to ieFile.

TIEImageListEx

Unit ImageEnView

TImageEnView

DECLARATION

```
TIELocation = (ieMemory, ieFile, ieTBitmap);
```

DESCRIPTION

Value	Description
ieMemory	Uses standard memory. Canvas not available. Used for fast and little images.
ieFile	Uses memory mapped files. Canvas not available. Used for big images.
ieTBitmap	Uses TBitmap VCL object. Canvas available. Used for drawing and compatibility. Location can be assigned before or after Allocate. Assigning Location on an existing image it converts the image to new location.

The bitmap of a TIEImageListEx can be accessed by TIEImageListEx.Image[iIndex].Location. The filename can be accessed by TIEImageListEx.Filename[iIndex].

TIEImageListEx

Unit ImageEnView
TImageEnView

The TIEImageListEx class is available in the zip file for this document. The TIEImageListEx class is an extension of TIEImageList that provides features not found in TIEImageList. In addition to providing a function to replace images in the list it also provides bitdepth: integer, bitspersample: integer, samplesperpixel: integer, dimensions: string and color: string properties. These few additional properties makes it easy to store information with each TIEBitmap in the list that can be used to restore bitdepth and color information for display purposes when doing an undo after a bitdepth conversion for example.

The extended properties serve as an IO.Params list which would not be available otherwise. The properties are not automatically set when creating the TIEImageListEx item. You have to specify the properties values when filling the TIEImageListEx items:

***Example*:**

```
{private}
var
FIEImageListEx: TIEImageListEx;
…
iIndex :=
FIEImageListEx.AppendImageRef(TIEBitmap.Create
   (iImageENIO.IEBitmap), iFileName);
FIEImageListEx.Filename[iIndex] := iFileName;
FIEImageListEx.BitsPerSample[iIndex] :=
   iImageENIO.Params.BitsPerSample;
IEImageListEx.SamplesPerPixel[iIndex] :=
   iImageENIO.Params.SamplesPerPixel;
```

TIEImageListEx

```
FIEImageListEx.BitDepth[iIndex] :=
  iImageENIO.Params.BitsPerSample *
  iImageENIO.Params.SamplesPerPixel;
```

Bear in mind, however; that if you **change** a TIEBitmap in any way change (resize, resample, crop or change the bitdepth for example) with TImageENView you have to **replace the IEBitmap in the FIEImageListEx as well as any FIEImageListEx properties**. Using crop as an example, if you crop an image in TImageENView then replace the image in TIEImageListEx with the cropped image as follows:

```
procedure TForm1.Crop1Click(Sender: TObject);
// Crop an image using TimageEnView with a
// TIEImageListEx
var
  iListItem: TListItem;
  iFileName: string;
  iDimensions: string;
begin
  if ListView1.ItemIndex <> -1 then
  begin
     ImageENView1.Proc.SaveUndoCaptioned('Crop '
     + IntToStr(ImageENView1.Proc.UndoCount +
     1));
     Undo1.Hint := 'Crop ' +
        IntToStr(ImageENView1.Proc.UndoCount +
        1);
     ImageEnView1.Proc.CropSel();
```

TIEImageListEx

```
      ListView1.Items.BeginUpdate;
      iListItem :=
        ListView1.Items.Item[ListView1.ItemIndex];
      iDimensions :=
        IntegerToString(ImageEnView1.Bitmap.Width)
         + ' x ' + IntegerToString(
         ImageEnView1.Bitmap.Height) + ' pixels';
      iListItem.SubItems[1] := iDimensions;
      ListView1.Items.EndUpdate;
      iFileName := iListItem.SubItems[0];
      // Replace the IEMemoryList
      TIEImageListItemEx image with
      ImageEnView1.IEBitmap
      FIEImageListEx.Replace(ImageEnView1.
        IEBitmap, ListView1.ItemIndex);
      // change the Dimensions property
      FIEImageListEx.Dimensions[ListView1.
        ItemIndex] := IntegerToString(
        ImageEnView1.IO.IEBitmap.Width) +
        ' pixels x ' +
        IntegerToString(ImageEnView1.IO.IEBitmap
       .Height) + ' pixels';
      // Remove the crop selection
      ImageEnView1.DeSelect;
      Undo1.Enabled := ImageEnView1.Proc.CanUndo;
      StatusBar1.Panels[2].Text := iDimensions;
      Undo1.Enabled := ImageEnView1.Proc.CanUndo;
  end;
end;
```

Unit ImageEnView

TImageEnView

TIEImageListEx

```
procedure TForm1.Undo1Click(Sender: TObject);
// undo the last change, replace the lists
// TIEBitmap and restore the displayed
// information
var
  iListItem: TListItem;
  iDimensions: string;
  iBitDepth: integer;
  iColor: string;
begin
  ImageEnView1.Proc.Undo();
  ImageEnView1.Proc.ClearUndo;
  // replace the image in the list
  FIEImageListEx.Replace(ImageEnView1.IEBitmap,
     ListView1.ItemIndex);
  iListItem := ListView1.Items.Item[ListView1.
               ItemIndex];
  iDimensions :=
     IntegerToString(ImageEnView1.Bitmap. Width)
     + ' x ' + IntegerToString(
     ImageEnView1.Bitmap.Height) + ' pixels';
  // Replace the dimensions property
  iListItem.SubItems[1] := iDimensions;
  if ImageEnView1.IEBitmap.HasAlphaChannel then
    iBitDepth := 32
  else
    iBitDepth := ImageEnView1.IEBitmap.BitCount;
```

TIEImageListEx

```
  if iBitDepth = 32 then
    iColor := 'RGBA 32-Bit'
  else
    iColor := 'RGB ' + IntToStr(iBitDepth) + '-
       bit';
  iListItem.SubItems[2] := iColor;
FIEImageListEx.BitDepth[ListView1.ItemIndex] :=
    iBitDepth;
FIEImageListEx.Color[ListView1.ItemIndex] :=
    iColor;
  Undo1.Enabled := ImageEnView1.Proc.CanUndo;
  Undo2.Enabled := ImageEnView1.Proc.CanUndo;
  StatusBar1.Panels[2].Text := iDimensions;
  StatusBar1.Panels[3].Text := iColor;
end;
```

When using small glyphs or icons the TIEImageList memory (ieMemory) use is probably not a concern. When storing large images, typically photographs from digital cameras or scanner output then set the storage location to ieFile. See the ImageEnBitDepthWithImageListEx.zip demo in the zip file containing this document which demonstrates using TIEImageListEx with TImageEnView.

Unit ImageEnView

TImageEnView

Methods and Properties

TIEImageListEx methods and properties are shown below.

Create	Free	AppendImageRef
Clear	BitDepth	BitsPerSample
Color	Dimensions	Filename
FillFromDirectory	Remove	Replace

DECLARATION

```
property Create;
```

DESCRIPTION

Creates the TIEImageListEx class. This is the same as the TIEImageList class.

Example

```
var
FIEImageListEx: TIEImageListEx;
begin
   FIEImageListEx := TIEImageListEx.Create;
end;
```

TIEImageListEx

Unit ImageEnView

TImageEnView

Declaration

```
property Free;
```

Description

Free the TIEImageListEx class. This is the same as the TIEImageList class.

```
// Free the multiframe imagelist
  if Assigned(FIEImageListEx) then
  begin
    FIEImageListEx.Clear;
    FIEImageListEx.Free;
  end;
```

Declaration

```
AppendImageRef(image: TIEBitmap; filename:
WideString): integer;
```

Description

Appends the specified image. This method doesn't copy the bitmap, but just takes ownership of the image object. This is the same as the TIEImageList class.

TIEImageListEx

Parameter	Description
image	Image to append.
filename	Filename of the image

DECLARATION

```
property BitDepth[idx: integer]: integer;
```

DESCRIPTION

BitDepth is a property containing a integer value representing the BitDepth of the TIEBitmap at index idx of the image in the list. This is the same value as IO.Params.BitDepth.

DECLARATION

```
property BitsPerSample[idx: integer]: integer;
```

DESCRIPTION

BitsPerSample is a property containing an integer value representing the BitsPerSample of the TIEBitmap at index idx of the image in the list. This is the same value as IO.Params.BitsPerSample..

Unit ImageEnView

TImageEnView

Declaration

```
property Color[idx: integer]: string;
```

Description

Color is a property containing a string value representing the colors of the TIEBitmap at idx. This string is generally set as RGBA32-Bit or RGB24-Bit or RGB8-Bit or RGB4-Bit, but may be set to any string value you choose to describe the color.

Declaration

```
property Dimensions[idx: integer]: string;
```

Description

Dimensions is a string value containing the Dimensions of the image as a string with an index of idx. This typically is set to 1,024 pixels x 768 pixels or any other string value that describes the dimensions of the image

Declaration

```
procedure Clear: string;
```

Description

Clear removes all images from the list. This is the same as the TIEImageList class.

TIEImageListEx

Unit ImageEnView

TImageEnView

DECLARATION

```
property Filename[idx: integer]: string;
```

DESCRIPTION

Filename represents a string value containing the filename of the specified image with an index of idx. This is the same as the TIEImageList class.

DECLARATION

```
procedure FillFromDirectory(const Directory:
WideString; Limit: integer=-1;
AllowUnknownFormats: boolean=false; const
ExcludeExtensions: WideString='';
DetectFileFormat: boolean=false; const
FilterMask: WideString='');
```

DESCRIPTION

FillFromDirectory automatically loads all known images inside Directory. This is the same as the TIEImageList class.

TIEImageListEx

Unit ImageEnView

TImageEnView

Parameter	Description
Directory	Directory where to search files.
Limit	Specifies the maximum number of images to load. -1 means no limit.
AllowUnknownFormats	If false (default) loads only known and supported file formats. Otherwise tries to load all files.
ExcludeExtensions	Contains a comma separated list of file extensions to discard (i.e. 'lyr,all,iev').
DetectFileFormat	If true then the image type is detected reading the header, otherwise ImageEn looks at filename extension.
DetectFileFormat	Contains a comma separated list of file extensions to include. Empty string means "all supported extensions".

TIEImageListEx

Unit ImageEnView
TImageEnView

DECLARATION

```
property Remove(idx: integer);
```

DESCRIPTION

Removes the specified image (TIEBitmap) with an index of idx from the list. This is the same as the TIEImageList class.

DECLARATION

```
property Replace(image: TIEBitmap; idx: integer);
```

DESCRIPTION

Replaces the image with an index of idx in the list with image (TIEBitmap).

DECLARATION

```
property SamplesPerPixel[idx: integer]: integer;
```

DESCRIPTION

SamplesPerPixel is an integer value containing the SamplesPerPixel of the image with an index of idx. This is the same value as IO.Params. SamplesPerPixel.

TIEImageListEx

Unit ImageEnView
TImageEnView

Drag and Drop Layers

Drag and drop can be used to create a layer containing the dropped image. To accomplish this add two TListview's. TimageEnView, two TImageLists, TTaskDialog, TStatusBar with 4 panels and a TImageEnProc to the form of a new project. Give one TListView a name of ListViewLayer1 and add 3 columns with a captions "#", "Name" and Dimensions. Give the other TListView a name of ObjectsListView1 and add one column with a caption of "Image Object". Set the DragMode property of the ObjectListView1 to dmAutomatic. Leave the DragMode property of ImageEnView1 set as dmManual. Give one TImageList a name of ObjectsImageList1 and the other LayersImageList1.

In the code editor add private five private variables: FFilePath: string; FObjectsFolder: string; FObjectBitmapToLarge: boolean; FObjectWidth: integer; FObjectHeight: integer;

Add the following code to the forms OnCreate event:

```
procedure TForm1.FormCreate(Sender: TObject);
var
  i: integer;
  iFilesList: TStringList;
  iCount: integer;
  iImageName: string;
  iPicture: TPicture;
  iBitmap: TBitmap;
  iListItem: TListItem;
  iFileType: integer;
  iExtension: string;
begin
```

Unit ImageEnView

TImageEnView

```
// create a bitmap with the dimensions of
// ImageEnViewImageEnView1.IEBitmap.Allocate(
ImageEnView1.Width,
ImageEnView1.Height);
ImageEnView1.IEBitmap.Fill(clWhite);
ImageEnView1.Update;
iFileType := ioPNG;
// See Helper Functions
iExtension := IEFileTypeToExtension(iFileType);
// See Helper Functions
FFilePath := DesktopFolder + 'Untitled' +
             iExtension;
Caption := 'ImageEnView Drag and Drop To A Layer
           -' + FFilePath;
StatusBar1.Panels[0].Text := ExtractFilePath(
                               FFilePath);
StatusBar1.Panels[1].Text := ExtractFilename(
                               FFilePath);
StatusBar1.Panels[2].Text := IntegerToString(
    ImageEnView1.Bitmap.Width) + ' pixels x ' +
    IntegerToString(
      ImageEnView1.Bitmap.Height) + ' pixels';
// Set the ImageEnView style
ImageEnView1.EnableAlphaChannel := True;
ImageEnView1.IO.Params.BMP_HandleTransparency :=
   True;
ImageEnView1.SetChessboardStyle(4, bsSolid);
// Allow selection of transparent layers
ImageEnView1.SelectionOptions :=
   ImageEnView1.SelectionOptions + [
   iesoSelectTranspLayers];
// The "object" images are located in child
// folder of the applications folder
```

Unit ImageEnView

TImageEnView

```
FObjectsFolder :=
   IncludeTrailingPathDelimiter ExtractFileDir(
      Application.ExeName)) + 'LargeObjects';
// This imagelist holds the "small object"
// images
ObjectsImageList1.Clear;
ObjectsListView1.Items.BeginUpdate;
try
  // create a list to hold the "object image file
// names"
iFilesList := TStringList.Create;
try
  if DirectoryExists(FLargeObjectsFolder) then
  begin
    // Add the images in the "Objects Folder" to
   // the fileslist and return a count of the
   // number of images
   // See Helper Functions
   iCount := GetAllFiles(FLargeObjectsFolder,
             '*.png', iFilesList);
   for i := 0 to iCount - 1 do
   begin
     if FileExists (iFilesList[i]) then
     begin
       iImageName := iFilesList[i];
       iPicture := TPicture.Create;
       try
         try
            iPicture.LoadFromFile(iImageName);
         except
            ShowMessage(iImageName + ' caused an
              exception and may be corrupt.  Try
              deleting this image and try
```

```
            starting this application again.');
          Break;
        end;
        // Draw the image in the imagelist to a
        // bitmap and add the bitmap to
        // ImageList1
        iBitmap := TBitmap.Create;
        try
          iBitmap.PixelFormat := pf32Bit;
          iBitmap.Width := 16;
          iBitmap.Height := 16;
          iBitmap.Canvas.StretchDraw(Rect(0, 0,
            16, 16), iPicture.Graphic);
          ObjectsImageList1.Add(iBitmap,
          iBitmap);
          // Create a Listview item
          iListItem :=
            ObjectsListView1.Items.Add;
          // Set the items caption
          iListItem.Caption := iImageName;
         // set the items imageindex
         iListItem.ImageIndex := i;
       finally
         iBitmap.Free;
       end;
     finally
       iPicture.Free;
     end;
   end
   else
   begin
     MessageBeep (MB_ICONERROR);
     TaskDialog1.MainIcon := tdiError;
```

```
  TaskDialog1.Title := 'Error';
  TaskDialog1.Caption := 'Folder Does Not Exist';
  TaskDialog1.Text := 'The folder ' +
     FLargeObjectsFolder + ' does not exist.';
  TaskDialog1.CommonButtons := [tcbOk];
  TaskDialog1.Execute;
end;
end;
end;
finally
   iFilesList.Free;
end;
finally
  ObjectsListView1.Items.EndUpdate;
end;
// display information
LabelObjectsCount1.Caption := 'Count: ' +
  IntegerToString(
    ObjectsListView1.Items.Count);
ShellListView1.RowSelect := True;
end;
```

Set the KeyPreview property of the form to True and add the following code to the OnFormKeyDown event:

```
procedure TForm1.FormKeyDown(Sender: TObject; var
Key: Word; Shift: TShiftState);
// Respond to the form's KeyDown Event and if the
// Delete key is pressed then delete the layer
var
  i: integer;
  ilayer: integer;
  iListItem: TListItem;
begin
  if Key = vk_Delete then
  begin
    // Get the layer to be deleted
    ilayer := ImageEnView1.LayersCurrent;
    if ilayer <> 0 then
    begin
      // Remove the layer from ImageEnView
      ImageEnView1.LayersRemove(ilayer);
      ImageEnView1.Update;
      // Update the layers ListView
      ListViewLayers1.Items.BeginUpdate;
      try
        ListViewLayers1.ItemIndex := ilayer - 1;
        iListItem := ListViewLayers1.Items[
          ilayer - 1];
        // Remove the item from ListViewLayers1
        iListItem.Delete;
        ListViewLayers1.Invalidate;
        // Remove the image from
        // cxImageListLayers1
        LayersImageList1.Delete(ilayer-1);
      finally
```

```
        ListViewLayers1.Items.EndUpdate;
      end;
    end;
    // Reset the Layers Listview image index
    // (because we deleted an image from the
    // ImageList2)
    for i := 0 to LayersImageList1.Count - 1
    do
    begin
      iListItem := ListViewLayers1.Items[i];
      iListItem.ImageIndex := i;
    end;
    if ImageEnView1.LayersCurrent > 0 then
    begin
      ListViewLayers1.ItemIndex :=
        ImageEnView1.LayersCurrent - 1;
      ListViewLayers1.Items.Item[
      ImageEnView1.LayersCurrent - 1].Selected :=
        True;
      if ListViewLayers1.CanFocus then
        ListViewLayers1.SetFocus;
    end;
  end;
end;
```

In the code editor add the following to the ImageEnView On DragDrop event:

Unit ImageEnView

TImageEnView

```
procedure TForm1.ImageEnView1DragDrop(Sender,
Source: TObject; X, Y: Integer);
var
  iIndex: integer;
  ilayer: integer;
  iListItem: TListItem;
  iDimensions: string;
  iObjectName: string;
  iBitmap: TBitmap;
begin
  Screen.Cursor := crHourGlass;
  try
    iIndex := ObjectsListView1.ItemIndex;
    iObjectName := JustName(
      ObjectsListView1.Items[iIndex].Caption);
    // Let ImageEn control the layers cursor
    ImageEnView1.Cursor := 1784;
    ImageEnView1.AutoCursors := True;
    ImageEnView1.IO.Params.BMP_HandleTransparency
       := True;
    // Add a layer
    ilayer := ImageEnView1.LayersAdd;
    // Set the layers position and dimensions and
    // properties
    ImageEnView1.CurrentLayer.name := AnsiString
     (iObjectName);
    ImageEnView1.CurrentLayer.PosX :=
       ImageEnView1.Layers[ilayer].ConvXScr2Bmp
        (X);
    ImageEnView1.CurrentLayer.PosY :=
    ImageEnniew1.Layers[ilayer].ConvYScr2Bmp
      (Y);
```

Unit ImageEnView

TImageEnView

```
ImageEnView1.CurrentLayer.VisibleBox := True;
ImageEnView1.CurrentLayer.Selectable := True;
ImageEnView1.LayersDrawBox := True;
// Add the layer info to the ListViewLayers
iListItem := ListViewLayers1.Items.Add;
iListItem.Caption := IntToStr(
   ListViewLayers1.Items.Count);
iListItem.SubItems.Add(iObjectName);
// Load the image
ImageEnView1.IO.LoadFromFile(
   ObjectsListView1.Items[iIndex].Caption);
FObjectWidth := ImageEnView1.Bitmap.Width;
FObjectHeight := ImageEnView1.Bitmap.Height;
iDimensions := IntegerToString(
   ImageEnView1.Bitmap.Width) + ' x ' +
   IntegerToString(ImageEnView1.Bitmap.Height);
    iListItem.SubItems.Add(iDimensions);
    iListItem.Selected := True;
// focus the selected layer and set its
// dimensions to the bitmap
ImageEnView1.Layers[ilayer].Width :=
   FObjectWidth;
ImageEnView1.Layers[ilayer].Height :=
   FObjectHeight;
 // Add a bitmap to the layers imagelist
 iBitmap := TBitmap.Create;
 try
   iBitmap.Width := 16;
   iBitmap.Height := 16;
   iBitmap.Assign(ImageEnView1.Bitmap);
   ImageEnProc1.AttachedBitmap := iBitmap;
   ImageEnProc1.Resample( 16, 16 );
   // ImageList2 holds the images for the
```

Unit ImageEnView

TImageEnView

```
//layers ListView
if LayersImageList1.AddMasked(iBitmap,
   clBlack ) <> -1 then
// update the frame imageindex to match the
// bitmap
iListItem.ImageIndex := ImageEnView1.LayersCount
   - 2;
finally
   iBitmap.Free;
 end;
ImageEnView1.Proc.SetTransparentColors(
ImageEnView1.Layers[ilayer].Bitmap.Pixels[0,
ImageEnView1.Layers[ilayer].Bitmap.Height
    - 1], ImageEnView1.Layers[ilayer
].Bitmap.Pixels [0,
          ImageEnView1.Layers[ilayer
].Bitmap.Height - 1], 0);
// Display layer info
SelectedLayer1.Caption := 'Selected Layer: ' +
IntToStr(ilayer);
LayerName1.Caption := 'Layer Name: ' + string (
ImageEnView1.CurrentLayer.name);
LayersCount1.Caption := 'Layer Count: ' +
IntToStr(ImageEnView1.LayersCount - 1);
// Allow selecting, moving and resizing the
// layers
ImageEnView1.MouseInteract := [miMoveLayers,
   miResizeLayers];
ImageEnView1.EnableAlphaChannel := True;
ImageEnView1.Bitmap.Modified := True;
ImageEnView1.Update;
finally
  Screen.Cursor := crDefault;
```

```
end;
end;
```

In the ImageEnView OnDrop event add the following code:

```
procedure TForm1.ImageEnView1DragOver(Sender,
Source: TObject; X, Y: Integer; State:
TDragState; var Accept: Boolean);
// Set the "Objects" Listview DragMode property
// to dmAutomatic in the IDE
var
  iIndex: integer;
  iBitmap: TBitmap;
begin
  if Source is TListView then
    Accept := True;
  iIndex := ListViewLayers1.ItemIndex;
  // Get the bitmap of the dropped object
  iBitmap := TBitmap.Create;
  try
    iBitmap.PixelFormat := pf32Bit;
    ObjectsImageList1.GetBitmap(iIndex,
      iBitmap);
  // Set FObjectBitmapToLarge if dropped bitmap
  // dimensions are less than Layer 0 bitmap
  // dimensions
  if ( iBitmap.Width <= ImageEnView1.Layers [0
     ].Bitmap.Width) and
       (iBitmap.Height <= ImageEnView1.Layers [0
     ].Bitmap.Height) then
       FObjectBitmapToLarge := False
    else
```

```
      FObjectBitmapToLarge := True;
  finally
    iBitmap.Free;
  end;
end;
```

In the ImageEnView LayerNotify event add the following code:

```
procedure TForm1.ImageEnView1LayerNotify(Sender:
TObject; layer: Integer; event: TIELayerEvent);
var
  iDimensions: string;
  iListItem: TListItem;
  iMS: TMemoryStream;
begin
  // Set the Layers Listview index biased on the
  // layer
  if layer <> 0 then
  begin
    SelectedLayer1.Caption := 'Selected Layer: '
      + IntToStr(layer);
    LayerName1.Caption := 'Layer Name: ' + string
      (ImageEnView1.CurrentLayer.name);
    LayersCount1.Caption := 'Layer Count: ' +
       IntToStr(ImageEnView1.LayersCount - 1);
    ListViewLayers1.Selected := nil;
    if layer > 0 then
      ListViewLayers1.ItemIndex := layer - 1
    else
      ListViewLayers1.ItemIndex := -1;
  end
```

```
  else
    SelectedLayer1.Caption := 'Selected Layer:';
    // If resizing the layer update the
    // dimensions in the listview
    if event = ielResizing then
    if layer > 0 then
    begin
      iListItem := ListViewLayers1.Items.Item[
         layer - 1];
      iDimensions := IntegerToString(
         ImageEnView1.Layers[layer
        ].Width) + ' x ' + IntegerToString
        ( ImageEnView1.Layers[layer].Height);
      iListItem.SubItems.Strings[1] :=
         iDimensions;
    end;
end;
```

In the ListViewLayers1 SelectItem event add the following code:

```
procedure TForm1.ListViewLayers1SelectItem(
Sender: TObject; Item: TListItem; Selected:
Boolean);
begin
  // If the ListViewLayers item is selected
  // update the info and selected layer
  if Selected then
  begin
    ImageEnView1.LayersCurrent := Item.Index + 1;
    SelectedLayer1.Caption := 'Selected Layer: '
       + IntToStr(Item.Index + 1);
```

```
    LayersCount1.Caption := 'Layers: ' + IntToStr
       (ListViewLayers1.Items.Count);
    LayerName1.Caption := 'Layer Name: ' +
       Item.SubItems[0];
  end
  else
  begin
    SelectedLayer1.Caption := 'Selected Layer: ';
    LayersCount1.Caption := 'Layers:' + IntToStr
       (ImageEnView1.LayersCount - 1);
    LayerName1.Caption := 'Layer Name: ';
  end;
end;
```

See the ImageEnViewLayersDragAndDrop Demo in the zip file containing this document to demonstrate the use of Drag & Drop layers, Merging layers and deleting ImageEnView layers and handling layers in a preview.

Unit ImageEnView
TImageEnView

Drawing with ImageEn

Although ImageENView does not have built-in drawing functions GDI and GDIPlus drawing ImageEnView is possible. Other third-party components can also be used with ImageENView for adding brush drawing to an application. Both GDI and GDIPlus drawing is done by adding code the ImageEnView.MouseDown, MouseMove and MouseUp ImageEnView events. GDI drawing can be done by using the ImageEnView.Bitmap .Canvas and GDI **or** GDIPlus drawing is accomplished by using a TIECanvas class. GDI drawing is done if

A significant amount of work (adding components and writing code in the component events as well as in the ImageEnView events) is required. TButtons or TSpeedButtons to control the type of drawing to be done, TSpinEdit's used to set the pen and brush sizes and to control the opacity of the brushes or lines, color controls such as TColorBox to select pen and brush colors must be added to the applications GUI interface. Once the controls have been added and code written to handle setting the GUI graphics drawing properties all that remains is to add code to the ImageEnView mouse events to do the actual drawing.

A simple TIECanvas drawing demo is available in the zip file containing this document.

Unit ImageEnView

TImageEnView

Handling the mouse events for IECanvas Drawing:

```
procedure TForm1.MyUndo(ie: TImageEnView);
var
  ix1, iy1, ix2, iy2: integer;
begin
  ix1 := startX;
  iy1 := startY;
  ix2 := lastX;
  iy2 := lastY;

  //OrdCor is defined in ImageEnProc
  OrdCor(ix1, iy1, ix2, iy2);
  ie.Proc.UndoRect(ix1 -
    StrToInt(BrushSize1.Text), iy1 -
    StrToInt(BrushSize1.Text), ix2 +
    StrToInt(BrushSize1.Text),
    iy2 + StrToInt(BrushSize1.Text));
end;
```

Unit ImageEnView

TImageEnView

```
procedure TForm1.ImageEnView1MouseDown(Sender:
TObject; Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
var
  ix1: integer;
  ix2: integer;
  iy2: integer;
  iy1: integer;
begin
  ImageEnView1.Proc.SaveUndo;
  StartX := ImageEnView1.XScr2Bmp(X);
  StartY := ImageEnView1.YScr2Bmp(Y);
  BrushSize := StrToIntDef(BrushSize1.Text, 20);
  PX := StartX - BrushSize div 2;
  PY := StartY - BrushSize div 2;
  mx := x; //StartX;
  my := y; //StartY;
  lx := StartX;
  ly := StartY;
  Drawing := False;
  if (Draw1.Down) then
  begin
    ImageEnView1.Proc.SaveUndoCaptioned('Draw ' +
      IntToStr(ImageEnView1.Proc.UndoCount + 1));
    Undo1.Hint := 'Draw ' +
      IntToStr(ImageEnView1.Proc.UndoCount + 1);
    Drawing := True;
    BrushSize := StrToIntDef(BrushSize1.Text,
      20);
    BrushColor := BrushColor1.Selected;
    BrushAlpha := StrToInt(BrushAlpha1.Text);
```

Unit ImageEnView

TImageEnView

```
// Important- if UseGDIPlus1 is checked then
// a GDIPlus canvas is used else a TCanvas is
// used instead
NewCanvas := TIECanvas.Create(ImageEnView1.
  Layers[ImageEnView1.LayersCurrent].
  Bitmap.Canvas, AntiAlias1.Checked,
  UseGDIPlus1.Checked);
MyUndo(ImageENView1);
ix1 := px;
iy1 := py;
ix2 := px + BrushSize;
iy2 := px + BrushSize;
//OrdCor is defined in ImageEnProc
OrdCor(ix1, iy1, ix2, iy2);
NewCanvas.Pen.Width := 1;
NewCanvas.Pen.Color := BrushColor;
NewCanvas.Pen.Transparency := BrushAlpha;
NewCanvas.Brush.Transparency := BrushAlpha;
NewCanvas.Brush.BackTransparency :=
  BrushAlpha;
NewCanvas.Brush.Color :=
  BrushColor1.Selected;
NewCanvas.Pen.Style := psSolid;
NewCanvas.Brush.Style := bsSolid;
NewCanvas.SmoothingMode :=
TIECanvasSmoothingMode(IECanvasSmoothingMode1
  .ItemIndex);
MyUndo(ImageENView1);
case PenType1.ItemIndex of
  0:
    NewCanvas.Ellipse(Rect(PX, PY, PX +
      BrushSize, PY + BrushSize));
```

```
      1:
        NewCanvas.FillRect(Rect(PX, PY, PX +
          BrushSize, PY + BrushSize));
    end;
    // Free the canvas
    //Important-
    NewCanvas.Free;
  end;
  ImageEnView1.Update
end;


procedure TForm1.ImageEnView1MouseMove(Sender:
TObject; Shift: TShiftState; X, Y: Integer);
var
  BX, BY: Integer;
begin
  BX := ImageEnView1.XScr2Bmp(X);
  BY := ImageEnView1.YScr2Bmp(Y);
  BrushSize := StrToIntDef(BrushSize1.Text, 20);
  PX := BX - BrushSize div 2;
  PY := BY - BrushSize div 2;
  mx := X;
  my := Y;
  lx := X;
  ly := Y;  if Draw1.Down then
  begin
    Drawing := True;
    ImageEnView1.Update;
  end
  else
```

```
  begin
    Drawing := False;
    ImageEnView1.Update;
  end;
  if (ImageEnView1.XScr2Bmp(X) >= 0) and
   (ImageEnView1.XScr2Bmp(X) <=
     ImageEnView1.IEBitmap.Width - 1) and
      (ImageEnView1.YScr2Bmp(Y) >= 0) and
        (ImageEnView1.YScr2Bmp(Y) <=
          ImageEnView1.IEBitmap.Height - 1) then
  begin
    StatusBar1.Panels[6].Text := 'X: ' +
      IntToStr(BX);
    StatusBar1.Panels[7].Text := 'Y: ' +
      IntToStr(BY);
    ColorUnderCursor1.Brush.Color :=
      ImageEnView1.IEBitmap.Canvas.Pixels
        [BX, BY];
    StatusBar1.Panels[8].Text := 'Color: ' +
      ColorToString(ColorUnderCursor1.Brush.
        Color);
  end;
  // Draw with ieCanvas
  if (Draw1.Down) and (Shift = [ssLeft]) then
  begin
    Drawing := True;
    BrushSize := StrToIntDef(BrushSize1.Text,
      20);
    BrushColor := BrushColor1.Selected;
    BrushAlpha := StrToInt(BrushAlpha1.Text);
    // Create a canvas
    // Important- if UseGDIPlus1 is checked then
    // a GDIPlus canvas is used else a TCanvas is
```

```
    // used instead
    NewCanvas := TIECanvas.Create(ImageEnView1.
      Layers[ImageEnView1.LayersCurrent].Bitmap.
        Canvas, AntiAlias1.Checked,
          UseGDIPlus1.Checked);
    NewCanvas.Pen.Width := 1;
    NewCanvas.Pen.Color := BrushColor;
    NewCanvas.Pen.Transparency := BrushAlpha;
    NewCanvas.Brush.Transparency := BrushAlpha;
      NewCanvas.Brush.BackTransparency :=
        BrushAlpha;
    NewCanvas.SmoothingMode :=
      TIECanvasSmoothingMode(
        IECanvasSmoothingMode1.ItemIndex);
    // draw
    case PenType1.ItemIndex of
      0:
        NewCanvas.Ellipse(Rect(PX, PY, PX +
          BrushSize, PY + BrushSize));
      1:
        NewCanvas.FillRect(Rect(PX, PY, PX +
          BrushSize, PY + BrushSize));
    end;
    // Free the canvas
    // Important- NewCanvas.Free;
  end;
  LastX := ImageEnView1.XScr2Bmp(X);
  LastY := ImageEnView1.YScr2Bmp(Y);
end;
```

Unit ImageEnView

TImageEnView

```
procedure TForm1.ImageEnView1MouseUp(Sender:
TObject; Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
begin
  ImageEnView1.Update;
end;
```

Drawing In The Backbuffer Event

BackBuffer gets access to the back buffer where ImageEn will draw the image (and layers) just before the paint event. You can draw on BackBuffer handling the OnDrawBackBuffer event to paint custom graphics on it.

You can draw on BackBuffer handling the OnDrawBackBuffer event to paint custom graphics on it. **BackBuffer** is updated whenever **Update** is called.

Any drawing in the OnBackBuffer event does not become a part of the Bitmap or IEBitmap. The OnBackBuffer event is a temporary drawing surface so the OnDrawBackBuffer event is the best place to draw with ImageEnView to show the position of the mouse as an ellipse or square for pixel editing purposes or for showing vertical and horizontal lines as an image rotate reference for example.

Unit ImageEnView
TImageEnView

Drawing in the BackBuffer may produce speed issues with images larger than 2,048x1,536 with a TResampleFilter. If BackBuffer drawing is too slow for your needs do not use layer filters with large pictures or get the coordinates for the drawing in the OnMouseDown event instead of in the OnMouseMove event. Without using a TResampleFilter drawing speed did not seem to be an issue least visually even with very large pictures. I do not know the image size limit when speed becomes an issue when drawing in the BackBuffer with TResampleFilter.

Unit ImageEnView
TImageEnView

Example

Drawing an ellipse in the BackBuffer Event

```
private
{ Private declarations }
// holds the backbuffer drawing coordinates
fmx, fmy: integer;

procedure TForm1.ImageEnView1MouseDown(Sender:
TObject; Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
// Get the coordinates for backbuffer drawing
// (useful for large images with
// TResampleFiltering)
begin
  fmx := X;
  fmy := Y;
  // Update calls OnDrawBackBuffer event
  ImageEnView1.Update;
end;


procedure TForm1.ImageEnView1MouseMove(Sender:
TObject; Shift: TShiftState; X, Y: integer);
var
begin
  // Get the coordinates for backbuffer drawing
  fmx := X;
  fmy := Y;
  // Update calls OnDrawBackBuffer event
  ImageEnView1.Update;
end;
```

Unit ImageEnView

TImageEnView

```
procedure TForm1.ImageEnViewDrawBackBuffer(
Sender: TObject);
var
  i: Double;
  z: Double;
  mx1: Double;
  my1: Double;
  lx1: Double;
  ly1: Double;
  x1, y1, x2, y2: Integer;
  bx, by: Integer;
begin
  if PaintBrush1.Down then
  begin
    // fmx and fmy set in ImageEnView1MouseMove
    bx := ImageEnView1.XScr2Bmp(fmx);
    by := ImageEnView1.YScr2Bmp(fmy);
    if (bx >= 0 ) and (bx <=
       ImageEnView1.IEBitmap.Width - 1) and (by
        >= 0 ) and (by <=
        ImageEnView1.IEBitmap.Height - 1) then
    begin
     with ImageEnView1.BackBuffer.Canvas do
      begin
        i := BrushPaint1.BrushSize / 2;
        Pen.Mode := pmCopy;
        { Make sure brush is visible regardless
        of the color under the cursor by setting
        pen color to a contrasting with the color
        under the cursor }
        Pen.Color := clRed;
        Brush.Style := bsClear;
        z := ImageEnView1.Zoom / 100;
```

```
        mx1 := fmx;
        my1 := fmy;
        lx1 := flx;
        ly1 := fly;
        x1 := Round(mx1 - i * z);
        y1 := Round(my1 - i * z);
        x2 := Round(lx1 + i * z);
        y2 := Round(ly1 + i * z);
        Ellipse(x1, y1, x2, y2);
      end;
    end;
  end;
end;
```

Unit ImageEnView

TImageEnView

Example

Drawing a horizontal and vertical line in the BackBuffer Event. This requires getting the X,Y coordinates in ImageEnViewMouseDown or ImageEnViewMouseMove.

```
procedure TForm1.ImageEnView1MouseDown(Sender:
TObject; Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
// Get the coordinates for backbuffer drawing
// (useful for large images when using
// TResampleFilter)
begin
  fmx := X;
  fmy := Y;
  // Update calls OnDrawBackBuffer event
  ImageEnView1.Update;
end;


procedure TForm1.ImageEnView1MouseMove(Sender:
TObject; Shift: TShiftState; X, Y: integer);
var
begin
  // Get the coordinates for backbuffer drawing
  fmx := X;
  fmy := Y;
  // Update calls OnDrawBackBuffer event
  ImageEnView1.Update;
end;
```

Unit ImageEnView

TImageEnView

```
procedure
TForm1.ImageEnView1DrawBackBuffer(Sender:
TObject);
// Draw crosshairs over the TImageEnView
// component.  This works at any zoom level and
// the crosshairs appear over the entire
component
begin
  with ImageEnView1.BackBuffer.Canvas do
  begin
    Pen.Mode := pmCopy;
    Pen.Width := 3;
    Pen.Color := clRed;
    Brush.Style := bsClear;
    MoveTo(0, fmy);
    LineTo(ImageEnView1.Width, fmy);
    MoveTo(fmx, 0);
    LineTo(fmx, ImageEnView1.Height);
  end;
end;
```

Rotating Small Images

When rotating images at other than simple angles (90, 180, 270) small images can be distorted by the rotation. To reduce the distortion you can select the anti-alias algorithm using AntialiasMode parameter. The AntialiasMode parameter can be:
ierFast : fast but with low quality
ierBilinear : bilinear, high quality
ierBicubic : bicubic, maximum quality

The best quality rotations with small images are usually produced with ierBicubic antialias mode.

 Original

 ierBicubic

 ierFast

Unit ImageEnView
TImageEnView

Methods and Properties

TImageEnView methods and properties have been split into various categories consisting of Alpha Channel, Display, Other, Cursor, Bitmap, Bitmap Alpha Channel, User Actions, Selections, Transitions, Layers, Scrollbars, Input/output, Processing, and Events.

Alpha Channel

AlphaChannel

HasAlphaChannel

EnableAlphaChannel

Unit ImageEnView
TImageEnView

Display

AutoShrink	AutoStretch	Center
CenterImage	DisplayGrid	DisplayImageRect
DelayTimer	DelayZoomFilter	DelayZoomTime
ExtentX	ExtentY	Fit
FitToHeight	FitToWidth	GetIdealZoom
GetMaxViewXY	GetRenderRectangles	IdealComponentHeight
IdealComponentWidth	LoadState	LockPaint
LockUpdate	OffScreenPaint	OffsetX
OffsetY	PaintRect	ResetState
SaveState	SetChessboardStyle	SetViewXY
SetViewXYSmooth	SmoothScrollValue	Stretch
UnLockPaint	UnLockUpdate	UnLockUpdateEx
Update	UpdateNoPaint	UpdateReason
UpdateRect	ViewX	ViewY
XBmp2Scr	XScr2Bmp	YBmp2Scr
YScr2Bmp	Zoom	ZoomAt
ZoomFilter	ZoomSelection	ZoomX
ZoomY		

Unit ImageEnView
TImageEnView

Navigator

IsNavigator

SetNavigator

Others – miscellaneous functions

GetCanvas	BackBuffer	BeginPostFrames
DoubleBuffer	DrawVersion	EndPostFrames
GetGripAt	HighlightedPixel	LockPaintCount
MoveContentTo		BackgroundStyle

Background

Background	GradientEndColor	WallPaperStyle
WallPaper		

Cursors

AutoCursors

Unit ImageEnView
TImageEnView

Bitmap- Bitmap Access

Bitmap and IEBitmap contain the image (current layer) to display. TIEBitmap is a replacement of VCL TBitmap class. The object TIEBitmap also contains the alpha channel of the image. TIEBitmap has many methods and properties compatible with TBitmap and enhances it supporting multi-threading and large images. TIEBitmap can store images in memory mapped files (for big images), in memory (fast access) or can encapsulate TBitmap objects (for canvas drawing and compatibility).

When you load or assign an image to Bitmap a corresponding IEBitmap is created which provides enhanced features such as access to an images alpha channel (32-bit images). Generally speaking when full alpha channel access is required in an application you should use IEBitmap access.

Bitmap

DpiX	DpiY	SetDPI
Bitmap	Blank	ChangeResolution
Clear	CopyFromPolygon	CopyToPolygon
DrawTo	IEBitmap	ImageChange
IsEmpty	LegacyBitmap	SetExternalBitmap
SetSelectedPixelsColor		

Unit ImageEnView

TImageEnView

TIEBitmap is a replacement of VCL TBitmap class. It has many methods and properties compatible with TBitmap and enhances it supporting multi-threading and large images. TIEBitmap can store images in memory mapped files (for big images), in memory (fast access) or can encapsulate TBitmap objects (for canvas drawing and compatibility).

IEBitmap

Access	Palette	PaletteLength
Allocate	Alpha	AlphaChannel
Assign	AssignImage	AutoCalcBWValues
BitAlignment	BitCount	BlackValue
CalcRAWSize	Canvas	CanvasCurrentAlpha
ChannelCount	ChannelOffset	Contrast
CopyAndConvertForm	CopyFromMemory	CopyFromDIB
CopyFromTBitmap	CopyFromTDibBitmap	CopyFromTIEMask
CopyPaletteTo	CopyRectTo	CopyToTBitmap
CopyToTDibBitmap	CopyToTIEMask	CopyWithMask1
CopyWithMask2	Create	CreateDIB
DefaultDitherMethod	Destroy	DrawToCanvas
EncapsulatedFromMe	EncapsulatedFromTBitmap	EncapsulateMemory
EncapsulateTBitmap	Fill	FillRect

Unit ImageEnView
TImageEnView

IEBitmap

FreeImage	FreeRow	Full
GetHash	GetRow	HasAlphaChannel
Height	IsAllBlack	IsEmpty
IsGrayScale	LoadRAWFromBufferOrStream	Location
MemoryAllocator	MergeAlphaRectTo	MergeFromTDibBitmap
MergeWithAlpha	MinFileSize	MoveRegion
Origin	PaletteUsed	PixelFormat
Pixels_ie16g	Pixels_ie1g	Pixels_ie24RGB
Pixels_ie32f	Pixels_ie32RGB	Pixels_ie48RGB
Pixels_ie8	Pixels_ieCIELab	Pixels_ieCMYK
Pixels	PPixels_ie24RGB	PPixels_ie32RGB
PPixels_ie48RGB	Read	RemoveAlphaChannel
RenderToCanvas	RenderToCanvasWithAlpha	RenderToTBitmap
RenderToTBitmapEx	RenderToTIEBitmap	RenderToTIEBitmapEx
Resize	Rowlen	SaveRAWToBufferOrStream
ScanLine	StretchRectTo	SwitchTo
SyncFull	SynchronizeRGBA	TBitmapScanlines
UndoInfo	UpdateFromTBitmap	VclBitmap
WhiteValue	Width	Write

Unit ImageEnView
TImageEnView

Bitmap Alpha Channel- bitmap alpha channel functions

CopyToBitmapWithAlpha

RemoveAlphaChannel

SetAlphaRangePixelsColor

SetSelectedAreaAlpha

User Actions- user functions

EnableShiftKey	EnableInteractionHints	ForceALTkey
AutoFixRotationBorders	MagicWandMaxFilter	MagicWandMode
MagicWandTolerance	MouseInteract	MouseScrollRate
MouseWheelParams	ZoomSelectionAspectRatio	

ImageEn selections consist of rectangular, ellipse, polygon and magic wand selections. The color of animated and unanimated selections may be controlled. Selection cursors are automatic as well.

Unit ImageEnView
TImageEnView

Selections- selection or rubberbanding

AddSelBreak	AddSelPoint	AssignSelTo
CopySelectionToBitmap	CopySelectionToIEBitmap	DelayDisplaySelection
DeSelect	DiscardSavedSelection	EndSelect
GetSelectionGripStyle	InvertSelection	IsPointInsideSelection
LoadSelectionFromFile	LoadSelectionFromStream	MakeSelectionFeather
MergeSelectionFromFile	MergeSelectionFromStream	MoveSelection
PolySel	PolySelCount	PolySelPoints
RestoreSelection	SavedSelectionsCount	SaveSelection
SaveSelectionToFile	SaveSelectionToStream	SelColor1
SelColor2	Select	SelectColors
SelectCustom	Selected	SelectedRect
SelectEllipse	SelectionAbsHeight	SelectionAbsWidth
SelectionAspectRatio	SelectionBase	SelectionGridSize
SelectionIntensity	SelectionMask	SelectionMaskDepth
SelectionOptions	SelectMagicWand	SelectRoundRect
SelX1	SelX2	SelY1
SelY2	SetSelectionGripStyle	SetSelectionMarkOuterStyle
VisibleSelection		

Unit ImageEnView
TImageEnView

Translations are visual image transform effects that displayed when running.

Transitions- image translation effects

AbortTransition	PrepareTransition
RunTransition	TransitionEndRect
TransitionRectMaintainAspectRatio	TransitionRunning
TransitionStartRect	TransitionTiming

Unit ImageEnView
TImageEnView

ImageEnView supports layers with transparency control. Layers may be the same size or have defined sizes which appear on top of the background image (Layer 0). ImageEnView always has at least 1 layer- the background layer. Layers may be moveable and resizeable as well.

Layers- layer functions

CurrentLayer	FindLayerAt	Layers
LayersAdd	LayersAutoUndo	LayersClear
LayersCopyToAlpha	LayersCount	LayersCreateFromAlpha
LayersCreateFromClipboar	LayersCreateFromSelection	LayersCurrent
LayersDrawBox	LayersDrawTo	LayersFixBorders
LayersFixRotations	LayersFixSizes	LayersInsert
LayersLoadFromFile	LayersLoadFromStream	LayersMerge
LayersMergeAll	LayersMergeTo	LayersMove
LayersRemove	LayersResizeAspectRatio	LayersRotateStep
LayersRotationAntialias	LayersRotationFilter	LayersSaveToFile
LayersSaveToStream	LayersSelectConstrains	LayersSync
MaxLayerHeight	MaxLayerWidth	SetLayersBoxStyle
SetLayersGripStyle	SoftCrop	SoftCropValue

Unit ImageEnView
TImageEnView

Scroll bars- visual scrollbar handling

FlatScrollBars	HScrollBarParams
ScrollBarsAlwaysVisible	ScrollBars
VScrollBarParams	

Input/Output

IO – See ImageEnIO

Image Processing

Proc- See ImageEnProc

Animated *P*olygons- animated polygon functions

AnimPolygonAddPt	AnimPolygonClear
AnimPolygonDel	AnimPolygonNew

Unit ImageEnView
TImageEnView

Events- component events

OnBeforeSelectionChange	OnDrawBackBuffer	OnDrawBackground
OnDrawCanvas	OnDrawLayerBox	OnDrawLayerGrip
OnDrawPolygon	OnDShowEvent	OnDShowNewFrame
OnFinishWork	OnImageChange	OnLayerNotify
OnMouseEnter	OnMouseInResizingGrip	OnMouseInSel
OnMouseLeave	OnPaint	OnProgress
OnSaveUndo	OnSelectionChange	OnSelectionChanging
OnSetCursor	OnSpecialKey	OnTransitionPaint
OnTransitionStep	OnTransitionStop	OnViewChange
OnViewChanging	OnVirtualKey	OnZoomIn
OnZoomOut		

Unit ImageEnView
TImageEnView

Methods and Properties

Alpha channel

DECLARATION

```
property AlphaChannel: TIEBitmap;
```

DESCRIPTION

Some formats like GIF, PNG, PSD, TIFF, ICO, CUR and TGA contains an alpha channel that specifies the image's pixels transparency.
The alpha channel is stored in the AlphaChannel property and in IEBitmap.AlphaChannel property.

DECLARATION

```
property EnableAlphaChannel: Boolean;
```

DESCRIPTION

If EnableAlphaChannel is true, ImageEn shows the image using the alpha channel. Some formats like Gif, Png, Tiff, Ico, Cur and Tga contain an alpha channel that specifies the image's pixels transparency. The alpha channel is stored in the AlphaChannel property (a TIEBitmap object) and in TIEBitmap.AlphaChannel property.

AlphaChannel

Unit ImageEnView

TImageEnView

DECLARATION

```
property HasAlphaChannel: Boolean;
```

DESCRIPTION

The property HasAlphaChannel returns true if the current image has an alpha channel.

Display

DECLARATION

```
property AutoShrink: Boolean;
```

DESCRIPTION

When AutoShrink is true TImageEnView follows these rules:
- If an image is bigger than TImageEnView window it is Shrink to fit;
- If an image is smaller than TImageEnView window it is displayed 100%

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
property AutoStretch: Boolean;
```

DESCRIPTION

When AutoStretch is true TImageEnView follows these rules:
- If an image is bigger than TImageEnView window it is displayed 100%;
- If an image is smaller than TImageEnView window it is stretched to fit

DECLARATION

```
property Center: Boolean;
```

DESCRIPTION

If Center is true, the image is centered the control's client area.

DECLARATION

```
procedure CenterImage;
```

DESCRIPTION

If the image is larger than the component client area, CenterImage will scroll the image making it centered.

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
property DisplayGrid: Boolean;
```

DESCRIPTION

When DisplayGrid is true and the Zoom is equal to or greater than 400, a grid is drawn. Each grid box is equivalent to a bitmap pixel.

Look also iegMINZOOMDISPLAYGRID.

DECLARATION

```
procedure DisplayImageRect(ARect: TRect);
```

DESCRIPTION

DisplayImageRect shows the specified image rectangle adjusting zoom and pan. The rectangle will be adjusted to get right aspect ratio.

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
property DelayTimer: integer;
```

DESCRIPTION

ImageEn has a timer that decrements a counter at each tick (you can set the tick delay using DelayTimer property). This timer controls the selection animation and the application of filters on scrolling (when DelayZoomFilter is true). If you set negative values DelayTimer changes its behavior. Negative values represent the maximum CPU time that ImageEn can use to show selections. For example setting: ImageEnView.DelayTimer:=-10; ImageEn will not use much more of 10% of the CPU time. The default DelayTimer's value is -20 (maximum 20% of the CPU time).

DECLARATION

```
property DelayZoomFilter: Boolean
```

DESCRIPTION

If DelayZoomFilter is true, the filter is applied with delay. This allows you to quickly navigate the image (zoom and scroll) and only after you have finished the navigation is the filter (quality zoom) is applied.

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
property DelayZoomTime: integer;
```

DESCRIPTION

ImageEn has a timer that decrements a counter at each tick (you can set the tick delay using DelayTimer property). This timer controls the selection animation and the application of filters on scrolling (when DelayZoomFilter is true). DelayZoomTime controls only the delay zoom filter application. The default value for this property is 4 (apply the filter after 4 ticks since last scroll).

DECLARATION

```
property DisplayGridLyr: integer;
```

DESCRIPTION

Specifies which layers to draw the grid.
0: Display the grid on all layers

- 1: Display the grid on the current layer (default behavior)
- 0: Display the grid on a specific layer See also iegMINZOOMDISPLAYGRID. See also DisplayGrid.

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
property ExtentX: integer;
```

DESCRIPTION

ExtentX is the width of the area used to show current image. See also ExtentY.

Read-only

DECLARATION

```
property ExtentY: integer;
```

DESCRIPTION

ExtentY is the height of the area used to show the current image. See also ExtentX.

Read-only

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure Fit;
```

DESCRIPTION

This method adjusts Zoom so that the image fits (with aspect ratio) the client area of the component.

DECLARATION

```
procedure FitToHeight;
```

DESCRIPTION

FitToHeight resizes the image (zoom) to fit the height of the control.

DECLARATION

```
procedure FitToWidth;
```

DESCRIPTION

FitToWidth resizes the image (zoom) to fit the width of the control.

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
function GetIdealZoom: double;
```

DESCRIPTION

GetIdealZoom returns the best zoom value to stretch the image in the component.

DECLARATION

```
procedure GetMaxViewXY(var mx, my: integer);
```

DESCRIPTION

Returns the maximum values for ViewX and ViewY (mx and my).

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure GetRenderRectangles(var xDst, yDst,
dxDst, dyDst: integer; var xSrc, ySrc, dxSrc,
dySrc: integer);
```

DESCRIPTION

Returns the rendered rectangle related to the client area and the source rectangle related to the bitmap of current layer.

xDst, yDst, dxDst, dyDst : the destination x, y and width, height where the image has been rendered

xSrc, ySrc, dxSrc, dySrc : the source x, y and width, height of the source image.

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
property IdealComponentHeight: integer;
```

DESCRIPTION

IdealComponentHeight is the height the image should have to fit exactly in TImageEnView component.

Example

```
// Resize ImageEnView to fully contain
// Carlotta.jpg
ImageEnView1.LoadFromFile('Carlotta.jpg');
ImageEnView1.Width :=
ImageEnView1.IdealComponentWidth;
ImageEnView1.Height :=
ImageEnView1.IdealComponentHeight;
```

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
property IdealComponentWidth: integer;
```

DESCRIPTION

IdealComponentWidth is the width the image should have to fit exactly in TImageEnView component.

Read-only

Example

```
// Resize ImageEnView to fully contain
// Carlotta.jpg
ImageEnView1.LoadFromFile('Carlotta.jpg');
ImageEnView1.Width :=
ImageEnView1.IdealComponentWidth;
ImageEnView1.Height :=
ImageEnView1.IdealComponentHeight;
```

DECLARATION

```
procedure LoadState(const FileName: string);
procedure LoadState(Stream: TStream);
```

DESCRIPTION

LoadState loads layers, selection and some other parameters like Zoom and Scroll position. Look also SaveState method.

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure LockPaint;
```

DESCRIPTION

The LockPaint method increases the lock counter's value. See also UnLockPaint.

Example

```
ImageEnView1.LockPaint;
ImageEnView1.ViewX := 10;
ImageEnView1.ViewY := 20;
ImageEnView1.Zoom := 300;
ImageEnView1.UnLockPaint;
```

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure LockUpdate;
```

DESCRIPTION

LockUpdate disables all calls to the Update method. LockUpdate and UnLockUpdate are useful, for example, when resampling multiple layers.

Example

```
ImageEnView1.LockUpdate;
for i:=0 to ImageEnView1.LayersCount-1 do begin
  ImageEnView1.LayersCurrent:=i;
  ImageEnView1.Proc.Resample(300,-1,rfNone);
  // without LockUpdate this call resizes all
  // other layers
end;
ImageEnView1.UnLockUpdate;
```

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
property OffScreenPaint: Boolean;
```

DESCRIPTION

When the visible property is false (the component is not visible) the back buffer BackBuffer is not painted. If you want the back buffer is correctly painted set OffScreenPaint=true.

DECLARATION

```
property OffsetX: integer;
```

DESCRIPTION

OffsetX gets the column where the image is shown.
If Center property is False, OffsetX will be 0.

See also OffsetY.

Read-only

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
property OffsetY: integer;
```

DESCRIPTION

OffsetY specifies the row where the image is shown.
If Center property is False, OffsetY will be 0.

See also OffsetX.

Read-only

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure PaintRect(const rc: TRect);
```

DESCRIPTION

PaintRect repaints only the rectangle rc without waiting for the Windows Paint message.

Example

```
// this code paints an image (width=50 height=50)
// at bitmap coordinates 10,10
ImageEnView1.Bitmap.Canvas.Draw(10,10, MyBitmap);
// map the bitmap coordinates to view coordinates
// (rc is a TRect type)
with ImageEnView1 do
  rc := Rect( XBmp2Scr(10), YBmp2Scr(10),
XBmp2Scr(10+50), YBmp2Scr(10+50) );
// update internal ImageEn state
ImageEnView1.UpdateRect(rc);
// paint the changes on the screen
ImageEnView1.PaintRect(rc);
```

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure ResetState();
```

DESCRIPTION

ResetState resets some TImageEnView properties to the default settings. In details ResetState() removes selection, empties layers, sets input/output parameters to the defaults, resets Zoom and Scroll and removes Undo's. Leaves LegacyBitmap unchanged.

DECLARATION

```
procedure SaveState(const FileName:string);
procedure SaveState(Stream: TStream);
```

DESCRIPTION

SaveState saves layers, selection and some other parameters like Zoom and Scroll position. Look also LoadState method.

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure SetChessboardStyle(Size: integer;
BrushStyle: TBrushStyle);
```

DESCRIPTION

SetChessboardStyle specifies size and brush of the chessboard background (see BackgroundStyle). Size specifies the box size (default 16). BrushStyle specifies the brush style of the boxes (default bsSolid).

DECLARATION

```
procedure SetViewXY(x, y: integer);
```

DESCRIPTION

SetViewXY sets ViewX and ViewY in one step.

DECLARATION

```
procedure SetViewXYSmooth(x,y: integer);
```

DESCRIPTION

SetViewXYSmooth sets ViewX and ViewY in one step, using smooth scrolling. Use SmoothScrollValue to control smooth.

Unit ImageEnView
TImageEnView

DECLARATION

```
property SmoothScrollValue: integer;
```

DESCRIPTION

Specifies the scrolling smooth when SetViewXYSmooth is called or when MouseInteract contains miMovingScroll. Large values increase smooth. “0” disables smooth (acts like SetViewXY). Default is 8.

DECLARATION

```
procedure Stretch;
```

DESCRIPTION

This method sets ZoomX and ZoomY values to stretch the image inside component borders. Using this method you loss the image aspect ratio and some functions, which require aspect ratio, may not properly work.

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
function UnLockPaint: integer;
```

DESCRIPTION

Use the UnLockPaint method to decrease the lock counter's value. If the lock count is zero, then the Update method is called. UnLockPaint returns the lock count. See also LockPaint

Example

```
ImageEnView1.LockPaint;
ImageEnView1.ViewX:=10;
ImageEnView1.ViewY:=20;
ImageEnView1.Zoom:=300;
ImageEnView1.UnLockPaint;
```

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure UnLockUpdate;
```

DESCRIPTION

UnLockUpdate re-Enables all calls to the Update method, after a call to LockUpdate. LockUpdate and UnLockUpdate are useful, for example, when you need to resample multiple layers. UnLockUpdate calls Update when the update can be unlocked.

Example

```
ImageEnView1.LockUpdate;
for i:=0 to ImageEnView1.LayersCount-1 do begin
  ImageEnView1.LayersCurrent:=i;
  ImageEnView1.Proc.Resample(300,-1,rfNone);
  // without LockUpdate this call resizes all
  // other layers
end;
ImageEnView1.UnLockUpdate;
```

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure UnLockUpdateEx;
```

DESCRIPTION

UnLockUpdateEx re-Enables all calls to the Update method, after a call to LockUpdate. LockUpdate and UnLockUpdateEx are useful, for example, when you need to resample multiple layers. UnLockUpdateEx doesn't call Update when the update can be unlocked.

DECLARATION

```
procedure Update;
```

DESCRIPTION

Update method updates the TImageEnView component with the actual image and redraws the client area. You have to call Update method whenever the Bitmap or IEBitmap property is modified in your code. Update is called automatically when an image processing or input operation is executed.

Example

```
// Draw a rectangle on the image and show it
ImageEnView1.Bitmap.Canvas.Rectangle(5,5,100,100);
ImageEnView1.Update;
```

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure UpdateNoPaint;
```

DESCRIPTION

Update method updates the TImageEnView component with the actual image. You have to call Update or UpdateNoPaint methods whenever the bitmap field is modified. UpdateNoPaint doesn't paint the image, but just refreshes the bitmap info.

DECLARATION

```
property UpdateReason: TIEUpdateReason;
```

DESCRIPTION

UpdateReason specifies the reason of next Update. This will help to optimize the Update job. This property is restored to ieurDefault after Update has been executed.

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure UpdateRect(rclip: TRect);
```

DESCRIPTION

UpdateRect updates the rectangle rclip. Use this function instead of Update when Bitmap objects change.

Example

```
// we assume that Zoom=100
ImageEnView1.Bitmap.Canvas.Fill(0,0,10,10);
ImageEnView1.UpdateRect(Rect(0,0,10,10));
```

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
property ViewX: integer;
```

DESCRIPTION

ViewX and ViewY are the first column and the first line to display. Use ViewX/Y instead of scroll bars to scroll images bigger than TImageEnView component sizes.

Example

```
// view from column 20 of bitmap
ImageEnView1.ViewX := 20;
// zoom image to 200%
ImageEnView1.Zoom := 200;
// view from column 40 of bitmap
ImageEnView1.ViewX := 20;
```

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
property ViewY: integer;
```

DESCRIPTION

ViewX and ViewY are the first column and the first line to display. Use ViewX/Y instead of scroll bars to scroll images bigger than TImageEnView component sizes.

Example

```
// view from column 20 of bitmap
ImageEnView1.ViewX := 20;
// zoom image to 200%
ImageEnView1.Zoom := 200;
// view from column 40 of bitmap
ImageEnView1.ViewX := 20;
```

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
function XBmp2Scr(x: integer):integer;
```

DESCRIPTION

YBmp2Scr or XBmp2Scr methods convert a bitmap coordinate to the window coordinate (applying Zoom and View status). x and y are bitmap coordinates. Returns a window coordinate. These methods are useful to select a bitmap region independently from the Zoom, ViewX and ViewY properties. If you are using multiple layers please use TImageEnView Layers[].ConvXBmp2Scr method instead.

Example

```
// Select Rect 10,10,100,100 of bitmap
ImageEnView1.Zoom(300);
x1 := XBmp2Scr(10);
y1 := YBmp2Scr(10);
x2 := XBmp2Scr(100);
y2 := YBmp2Scr(100);
ImageEnView1.Select( x1,y1,x2,y2);

// Select Rect 10,10,100,100 of the window
ImageEnView1.Select(10,10,100,100)
```

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
function XScr2Bmp(x: integer): integer;
```

DESCRIPTION

This method converts a window coordinate to the corresponding bitmap coordinate (applying Zoom and ViewX, ViewY status). x is a window coordinate. XSrc2Bmp returns a bitmap coordinate which can be negative or larger than bitmap size if the window coordinate is beyond the displayed image's boundaries. If you are using multiple layers please use TImageEnView.Layers[].ConvXScr2Bmp method instead.

Example

```
// X and Y are MOUSE coordinates
bx := ImageEnView1.XScr2Bmp(X);
by := ImageEnView1.YScr2Bmp(Y);
// now bx and by are Bitmap coordinates (referred
// to the ImageEnView1.Bitmap)
```

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
function YBmp2Scr(y: integer):integer;
```

DESCRIPTION

YBmp2Scr or XBmp2Scr methods convert a bitmap coordinate to the window coordinate (applying Zoom and View status). x and y are bitmap coordinates.

Returns a window coordinate. These methods are useful to select a bitmap region independently from the Zoom, ViewX and ViewY properties. If you are using multiple layers please use TImageEnView.Layers[].ConvYBmp2Scr method instead.

Example

```
    // Select Rect 10,10,100,100 of bitmap
ImageEnView1.Zoom(300);
x1 := XBmp2Scr(10);
y1 := YBmp2Scr(10);
x2 := XBmp2Scr(100);
y2 := YBmp2Scr(100);
ImageEnView1.Select( x1,y1,x2,y2);

// Select Rect 10,10,100,100 of the window
ImageEnView1.Select(10,10,100,100)
```

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
function YScr2Bmp(y: integer): integer;
```

DESCRIPTION

This method converts a window coordinate to the corresponding bitmap coordinate (applying Zoom and ViewX, ViewY status). y is a window coordinate. YScr2Bmp returns a bitmap coordinate which can be negative or larger than bitmap size if the window coordinate is beyond the displayed image's boundaries. If you are using multiple layers please use ImageEnView.Layers[].ConvYScr2Bmp method instead.

Example

```
// X and Y are MOUSE coordinates
bx := ImageEnView1.XScr2Bmp(X);
by := ImageEnView1.YScr2Bmp(Y);
// now bx and by are Bitmap coordinates (referred
// to the ImageEnView1.Bitmap)
```

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
property Zoom: double;
```

DESCRIPTION

Use the Zoom property to zoom the image in or out. No modifications are made to the image. Zoom is expressed in 100*percentage of the zoom. Zoom values less than 100 decreases the displayed size of the image and values greater than 100 increase the displayed size of the image.

See also

ZoomX
ZoomY

Example

```
// 2 pixels of image is 1 pixel of displayed
// video
ImageEnView1.Zoom:=50;

// 1 pixel of image is 2 pixels of view
ImageEnView1.Zoom:=200;

// floating point zoom
ImageEnView1.Zoom:=30.25;
```

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure ZoomAt(x, y: integer; ZoomVal: double;
Center: Boolean = true);
```

DESCRIPTION

This method zooms by the ZoomVal percentage centered at the point x,y. If the optional Center parameter if False, the zooming center is x, y, otherwise (the default) it is the center of image.

DECLARATION

```
property ZoomFilter: TResampleFilter;
```

DESCRIPTION

ZoomFilter specifies the filter to apply in zoom-in (Zoom property) operations. The fastest way to zoom a picture is to set ZoomFilter to rfNone (default).

Display

Unit ImageEnView

TImageEnView

DECLARATION

```
TResampleFilter = (rfNone, rfTriangle, rfHermite,
rfBell, rfBSpline, rfLanczos3, rfMitchell,
rfNearest, rfLinear, rfFastLinear, rfBilinear,
rfBicubic, rfProjectBW, rfProjectWB);
```

DESCRIPTION

If you need the best quality we suggest: rfHermite, rfBell, rfBSpline, rfLanczos3, rfMitchell, rfNearest, rfBilinear, rfBicubic. If you need speed we suggest: rfTriangle, rfLinear, rfFastLinear.

For projects (white on black or black on white) we suggest: rfProjectBW and rfProjectWB

Example

```
// Performs a 400% quality zoom
ImageEnView1.ZoomFilter:=rfFastLinear;
ImageEnView1.Zoom:=400;
```

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure ZoomSelection(AspectRatio: Boolean);
```

DESCRIPTION

The ZoomSelection method zooms to a rectangular area specified with Select method. If the optional parameter (default=true) AspectRatio is False the image area is stretched to the component sizes.

Example

```
ImageEnView1.Select(10,10,100,100);
ImageEnView1.ZoomSelection; // fill client area
(if possible) with rectangle 10,10,100,100
```

DECLARATION

```
property ZoomX: double;
```

DESCRIPTION

ZoomX specifies the horizontal Zoom. Using this property (and/or ZoomY) you loss the image aspect ratio and some functions, which require aspect ratio, may not properly work. Setting ZoomX and ZoomY to the same value is equivalent to set Zoom.

See also

Zoom
ZoomY

Display

Unit ImageEnView
TImageEnView

DECLARATION

```
property ZoomY: double;
```

DESCRIPTION

ZoomY specifies the vertical zoom. Using this property (and/or ZoomX) you loss the image aspect ratio and some functions, which require aspect ratio, may not properly work. Setting ZoomX and ZoomY to the same value is equivalent to set Zoom.

See also
Zoom
ZoomX

Display

Unit ImageEnView

TImageEnView

Navigator

DECLARATION

```
property IsNavigator: Boolean;
```

DESCRIPTION

Returns true if this object is used has navigator of another TImageEnView object.

See also

SetNavigator.

DECLARATION

```
procedure SetNavigator(nav: TImageEnView;
options: TIENavigatorOptions);
```

DESCRIPTION

SetNavigator specifies a TImageEnView component which works as navigator of current image. A navigator shows a selection that controls the zoom and scroll of the main control. Use options to fine tuning navigator behavior. Look at 'navigator' example for more info.

Navigator

Unit ImageEnView
TImageEnView

Example

```
ImageEnView1.SetNavigator(ImageEnView2,
[ienoMOUSEWHEELZOOM, ienoMARKOUTER]);
ImageEnView1.IO.LoadFromFile('input.jpg');
```

Important:

After calling SetNavigator do **NOT** attempt to update ImageEnView in other ways such as ImageEnView2.Assign(ImageEnView1) or ImageEnView2.Bitmap.Assign(ImageEnView1.Bitmap). Assignment to TImageEnView after calling SetNavigator usually causes exceptions.

Navigator

Unit ImageEnView
TImageEnView

Others

DECLARATION

```
function GetCanvas: TCanvas;
```

DESCRIPTION

The GetCanvas method returns the TImageEnView TCanvas object. Use it when you override Paint method to draw your object to the client area of the component.

Unit ImageEnView

TImageEnView

DECLARATION

```
property BackBuffer: TBitmap;
```

DESCRIPTION

BackBuffer gets access to the back buffer where ImageEn will draw the image (and layers) just before the paint event. You can draw on BackBuffer handling the OnDrawBackBuffer event to paint custom graphics on it.

BackBuffer is updated whenever Update is called.

Example

```
procedure ImageEnView1.OnDrawBackBuffer(Sender:
TObject);
begin
  with ImageEnView1.BackBuffer.Canvas do begin
    Pen.Color:=clRed;
    MoveTo( 0,0 );
    LineTo( 100,100 );
  end;
end;
```

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure BeginPostFrames(target: TImageEnView;
delay: integer; interval: integer);
```

DESCRIPTION

BeginPostFrames sends current image to a target TImageEnView, after “delay” milliseconds and then each “interval” milliseconds. This is useful to display captured frames to another control with delay. Look at capture\directshow5” example. You can (but you don’t need to do) end frame sending calling EndPostFrames.

Multiple calls to BeginPostFrames are possible.

Example

```
// sends current image to ImageEnView2 after 5
// seconds, and to ImageEnView3 after 10 seconds,
// at 50 ms for each frame
ImageEnView1.BeginPostFrames(ImageEnView2,5000,50);
ImageEnView1.BeginPostFrames(ImageEnView3,10000,50);
```

DECLARATION

```
property DoubleBuffer: Boolean;
```

DESCRIPTION

Not used. Always True.

Others

Unit ImageEnView
TImageEnView

DECLARATION

```
property DrawVersion: Boolean;
```

DESCRIPTION

When DrawVersion is true, displays the ImageEn version and release date on bottom-left side of the client area of the ImageEn component.

DECLARATION

```
procedure EndPostFrames(target: TImageEnView);
```

DESCRIPTION

EndPostFrames stops to send current image to target TImageEnView. Use BeginPostFrames to begin frame sending.

Example

```
ImageEnView1.EndPostFrames(ImageEnView2);
ImageEnView1.EndPostFrames(ImageEnView3);
```

Others

Unit ImageEnView
TImageEnView

DECLARATION

```
function GetGripAt(x,y: integer): TIEGrip;
```

DESCRIPTION

GetGripAt returns the selection grip at mouse position specified by x, y.

DECLARATION

```
property HighlightedPixel: TPoint;
```

DESCRIPTION

When coordinates are >-1 then a colored box is painted around the specified pixel.

Demo
viewers\imagecomp

Others

Unit ImageEnView

TImageEnView

DECLARATION

```
property LockPaintCount: integer;
```

DESCRIPTION

LockPaintCount returns lock painting state. 0 = no lock, > 0 locking. LockPaint increases LockPaintCount, UnLockPaint decreases it.

Read only

DECLARATION

```
procedure MoveContentTo(Destination:
TImageEnView);
```

DESCRIPTION

MoveContentTo transfers current image, all layers and input/output parameters to destination TImageEnView component. This method doesn't copy images, but just transfer pointers to images buffer. Source component will be empty, and Destination previous images will be removed.

Unit ImageEnView

TImageEnView

Background

DECLARATION

```
property BackgroundStyle: TIEBackgroundStyle;
```

DESCRIPTION

BackgroundStyle specifies the background style. The background is the component region not filled by the image or thumbnails.

See also

Background
GradientEndColor
WallPaper
WallPaperStyle

Example

```
ImageEnView1.BackgroundStyle := iebsPhotoLike;
```

Unit ImageEnView

TImageEnView

Declaration

TIEBackgroundStyle = (iebsSolid, iebsHorizontal, iebsVertical, iebsFDiagonal, iebsBDiagonal, iebsCross, iebsDiagCross, iebsChessboard, iebsDiagonals, iebsCropped, iebsCropShadow, iebsGradient, iebsSoftShadow, iebsPhotoLike);

Description

Value	Description
iebsCropped	3d border around the image (instead of around the component). We suggest that BorderStyle is bsNone and Center is true.
iebsCropShadow	shadow around the image
iebsChessboard	a chessboard picture
iebsDiagonals	two diagonals
iebsSolid	like bsSolid of TBrushStyle
iebsHorizontal	like bsHorizontal of TBrushStyle
iebsVertical	like bsVertical of TBrushStyle
iebsFDiagonal	like bsFDiagonal of TBrushStyle
iebsBDiagonal	like bsBDiagonal of TBrushStyle
iebsCross	like bsCross of TBrushStyle
iebsDiagCross	like bsDiagCross of TBrushStyle

Unit ImageEnView

TImageEnView

Value	Description
iebsGradient	a gradient (only Windows98/Me/2000/Xp)
iebsSoftShadow	soft (gaussian) shadow around the image
iebsPhotoLike	a chessboard inside the image with a black border around

Background

Unit ImageEnView
TImageEnView

DECLARATION

```
property Background: TColor;
```

DESCRIPTION

This property sets the background color. The background color is the color shown in unoccupied area when the current image is less than the control's visual size. This color is used also in geometric processing (such as rotation) to fill blank areas.

If you change Background color after the component has been created, at runtime, you don't see the change because the bitmap overlaps the background. For this reason, to change background color at runtime you have to:
- empty layer0 bitmap:
ImageEnView.Blank;
ImageEnView.Background := clRed;

- or fill the bitmap using the color you want:
ImageEnView.Proc.Fill(clRed);

See also
BackgroundStyle.

Example

```
// Rotate of 30 degrees and fill blank spaces
// with clBlack color
ImageEnView1.Background := clBlack;
ImageEnView1.Proc.Rotate(30, False);
```

Background

Unit ImageEnView
TImageEnView

DECLARATION

```
property GradientEndColor: TColor
```

DESCRIPTION

GradientEndColor specifies the ending color of the gradient when BackgroundStyle is iebsGradient.

DECLARATION

```
property WallPaperStyle: TIEWallPaperStyle;
```

DESCRIPTION

WallPaperStyle (defined in ImageEnView unit) specifies how to paint the wallpaper.

See also
WallPaper.

DECLARATION

```
property WallPaper: TPicture;
```

DESCRIPTION

The WallPaper property sets a background image under the main image and layers. Use WallPaperStyle to specify how to paint the wallpaper.

Background

Unit ImageEnView

TImageEnView

Cursors

DECLARATION

```
property AutoCursors: Boolean;
```

DESCRIPTION

Set AutoCursors to False if you want handle mouse cursors by hand. (Cursors loaded from a resource file.)

Bitmap

DECLARATION

```
property DpiX: integer;
```

DESCRIPTION

DpiX is the number of horizontal dots (pixels) per inch of the image. This value is loaded and saved to file in TImageEnIO component.

Bitmap

Unit ImageEnView
TImageEnView

DECLARATION

```
property DpiY: integer;
```

DESCRIPTION

DpiY is the number of vertical dots (pixels) per inch of the image. This value is loaded and saved to file in TImageEnIO component.

DECLARATION

```
procedure SetDPI(dpiX, dpiY: integer);
```

DESCRIPTION

The SetDPI method sets the horizontal (DpiX) and vertical (DpiY) dots per inch of the current image. These values will be saved together with the image.

DECLARATION

```
property Bitmap: TBitmap;
```

DESCRIPTION

Bitmap contains the image (current layer) to display. If LegacyBitmap is true, IEBitmap is just a wrapper for TBitmap that may be accessed using the Bitmap property.

Bitmap

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure Blank;
```

DESCRIPTION

Blank sets the image size to 1x1 (Width x Height).

Example

```
// load image
ImageEnView1.IO.LoadFromFile('image.tif');...
ImageEnView1.Blank; // free memory
```

DECLARATION

```
procedure ChangeResolution( NewDPI: integer;
ResampleFilter: TResampleFilter);
```

DESCRIPTION

ChangeResolution changes the DPI of the image, resampling it and setting new DPI values. Make sure source DPI contains valid values before call ChangeResolution.

Bitmap

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure Clear;
```

DESCRIPTION

Clear fills current image with Background color.

Example

```
ImageEn1.Background:=clWhite;
ImageEn1.Clear;
```

Bitmap

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure CopyFromPolygon(Source: TBitmap; const
Polygon: array of TPoint; PolygonLen: integer;
const Position: TPoint);
```

DESCRIPTION

CopyFromPolygon copies a region (Polygon) of source bitmap to current image. CopyFromPolygon enlarges current bitmap when needed.

Parameter	Description
Source	Source bitmap.
Polygon	Array of polygon vertexes (pixel coordinates related to Source bitmap).
PolygonLen	Number of vertexes in polygon.
Position	Destination point (pixel coordinate related to current image). The destination point is the top-left side of the rectangle that encloses source polygon

Bitmap

Example

```
// Copies current selected area of ImageEnView1
// to ImageEnView2 at position 0,0
ImageEnView2.CopyFromPolygon(ImageEnView1.Bitmap,
ImageEnView1.PolySelPoints^,ImageEnView1.PolySelC
ount,Point(0,0));
// Copies the Rect 0,0,100,100 in ImageEnView2 at
// position 0,0
ImageEnView2.CopyFromPolygon(ImageEnView1.Bitmap,
[Point(0,0),Point(100,0),Point(100,100),Point(0,1
00)],4,Point(0,0));
// Copies current selection to ImageEnView2. Then
// enhance contrast of ImageEnView2 and copy back
// to ImageEnView1 (in some selection).
ImageEnView2.CopyFromPolygon(ImageEnView1.Bitmap,
ImageEnView2.PolySelPoints^,ImageEnView2.PolySelC
ount,Point(0,0));
ImageEnView2.Proc.Contrast(30);
ImageEnView2.CopyToPolygon(ImageEnView1.Bitmap,Im
ageEnView1.PolySelPoints^,ImageEnView1.PolySelCou
nt,Point(0,0));
ImageEnView1.Update;
```

Bitmap

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure CopyToPolygon(Dest: TBitmap; const
Polygon: array of TPoint; PolygonLen: integer;
const Position: TPoint);
```

DESCRIPTION

CopyToPolygon copies a region (Polygon) of current image inside Dest bitmap. Polygon is an array of TPoint (pixel coordinates related to Dest bitmap) and PolygonLen is the number of point in Polygon. Position is the source point (pixel coordinate related to current image). The source point is the top-left side of the rectangle that encloses destination polygon.

CopyToPolygon enlarges destination bitmap when needed.

Example

```
// Copies current selection to ImageEnView2. Then
// enhance contrast of ImageEnView2 and copy back
// to ImageEnView1 (in some selection).
ImageEnView2.CopyFromPolygon(ImageEnView1.Bitmap,
ImageEnView2.PolySelPoints^,ImageEnView2.PolySelC
ount,Point(0,0));
ImageEnView2.Proc.Contrast(30);
ImageEnView2.CopyToPolygon(ImageEnView1.Bitmap,Im
ageEnView1.PolySelPoints^,ImageEnView1.PolySelCou
nt,Point(0,0));
ImageEnView1.Update;
```

Bitmap

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure DrawTo(Canvas: TCanvas);
```

DESCRIPTION

This method draws the current view in Canvas. The background isn't painted, and the image has up-left alignment.

Example

```
// draw in ImageEn2 all what is displayed in
// ImageEnView1
ImageEnView1.DrawTo(ImageEnView2.Bitmap.Canvas);
ImageEnView2.Update;
```

DECLARATION

```
property IEBitmap: TIEBitmap;
```

DESCRIPTION

IEBitmap contains the image (current layer) to display. The object TIEBitmap also contains the alpha channel of the image. If LegacyBitmap is True, IEBitmap is just a wrapper for TBitmap that may be accessed using the Bitmap property.

Bitmap

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure ImageChange; virtual;
```

DESCRIPTION

The ImageChange method generates an OnImageChange event.

DECLARATION

```
property IsEmpty: Boolean;
```

DESCRIPTION

IsEmpty returns true if the image is empty. At the moment IsEmpty just checks if the bitmap size is less than 2x2.

DECLARATION

```
property LegacyBitmap: Boolean;
```

DESCRIPTION

If LegacyBitmap is True, ImageEn uses TBitmap to store the image, otherwise ImageEn uses TIEBitmap. TIEBitmap handles images using memory mapped file (for large images) or main memory. This allows handling of large images and to include input/output and image processing in a multi-threaded environment. Also TIEBitmap supports a larger number of pixel formats (TIEPixelFormat).

Bitmap

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure SetExternalBitmap(Bmp: TIEBitmap);
```

DESCRIPTION

SetExternalBitmap makes it possible to connect a TImageEnView component to another one, sharing the same bitmap. This is useful to view the same image with multiple TImageEnView components, loading the image only one time.

Example

```
ImageEnView1.IO.LoadFromFile('input.jpg');
ImageEnView2.SetExternalBitmap(
ImageEnView1.IEBitmap );
//ImageEnView1 shows input.jpg. Also ImageEnView2
// shows the same image.
```

Bitmap

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure SetSelectedPixelsColor(Color: TRGB);
```

DESCRIPTION

SetSelectedPixelsColor sets selected pixels to the specified color.

Example

```
// select all pixels like pixel at (0,0), then
// fills all selected pixels using the White
// color.
ImageEnView.SelectMagicWand(0,0, iespReplace);
ImageEnView.SetSelectedPixelsColor(
CreateRGB(255,255,255) );
```

Bitmap

Unit ImageEnView

TImageEnView

Bitmap Alpha Channel

DECLARATION

```
procedure CopyToBitmapWithAlpha(Dest: TBitmap;
DestX, DestY: integer);
```

DESCRIPTION

CopyToBitmapWithAlpha copies the current image inside the Dest bitmap, at DestX, DestY position. If the image has an alpha channel, CopyToBitmapWithAlpha copies only visible area.

Example

```
ImageEnView1.CopyToBitmapWithAlpha(
ImageEnView2.Bitmap, 0,0);
ImageEnView2.Update;
```

Bitmap Alpha Channel

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure RemoveAlphaChannel(Merge: Boolean =
false);
```

DESCRIPTION

The RemoveAlphaChannel procedure removes alpha channel and the memory allocated for it. RemoveAlphaChannel sets HasAlphaChannel to false. If Merge is true the image will be merged with the background color using its alpha channel.

Example

```
// remove alpha channel from a gif
ImageEnView1.IO.LoadFromFile('test.gif');
ImageEnView1.RemoveAlphaChannel;
// this add a shadow (that uses the alpha
// channel) then save it to a jpeg (that cannot
// support alpha channel)
ImageEnView1.Proc.AddSoftShadow(4,3,3);
ImageEnView1.RemoveAlphaChannel( True );
// merge the alpha channel
ImageEnView.IO.SaveToFile('output.jpg');
```

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure SetAlphaRangePixelsColor(alphaMin,
alphaMax: integer; color: TRGB);
```

DESCRIPTION

SetAlphaRangePixelsColor sets all pixels that have an alpha channel value between alphaMin and alphaMax to the specified color.

Example

```
// this example loads an image with alpha
// channel, then paints all transparent pixels
// (0...254 alpha values) using White color,
// finally saves in a jpeg where we cannot save
// the alpha channel.
ImageEnView1.IO.LoadFromFile('test.png');
ImageEnView.SetAlphaRangePixelsColor(0,254,Create
RGB(255,255,255));
ImageEnView1.IO.SaveToFIle('output.jpg');
```

Bitmap Alpha Channel

Unit ImageEnView

TImageEnView

Declaration

```
procedure SetSelectedAreaAlpha(Alpha: integer);
```

Description

SetSelectedAreaAlpha sets the specified Alpha (alpha channel, transparency) value for all pixels inside current selection. To activate alpha channel (transparency), set EnableAlphaChannel to true.

Example

```
// set transparency to 180 inside 10,10,100,100
// rectangle
ImageEnView1.Select(10,10,100,100);
ImageEnVIew1.SetSelectedAreaAlpha(180);
// save with alpha channel
ImageEnView1.io.SaveToFile('image.png');
```

User Actions

Declaration

```
property EnableShiftKey: Boolean;
```

Description

EnableShiftKey enables or disables the shift key function (insert of multiple selections). The default is true.

Unit ImageEnView

TImageEnView

DECLARATION

```
property EnableInteractionHints: Boolean;
```

DESCRIPTION

When EnableInteractionHints is true (default) mouse interaction hints (layers rotation angle, movement position...) are enabled.

DECLARATION

```
property ForceALTkey: Boolean;
```

DESCRIPTION

Set ForceALTkey to true to emulate pressing the ALT key. It allows making and modifying selections or vectorial objects (for TImageEnVect component) always maintaining aspect ratio. Default is False.

DECLARATION

```
property AutoFixRotationBorders: Boolean;
```

DESCRIPTION

If AutoFixRotationBorders is true (default) ImageEn will automatically call LayersFixBorders method at the entering in rotation mode.

User Actions

Unit ImageEnView

TImageEnView

DECLARATION

```
property MagicWandMaxFilter: Boolean;
```

DESCRIPTION

Set MagicWandMaxFilter to true (default is False) to apply a maximum filter to the magic wand selection (remove black hole).

DECLARATION

```
property MagicWandMode: TIEMagicWandMode;
```

DESCRIPTION

MagicWandMode specifies how magic wand selection works. Default is iewInclusive.

Unit ImageEnView

TImageEnView

DECLARATION

TIEMagicWandMode=(iewInclusive, iewExclusive, iewGlobal);

DESCRIPTION

Value	Description
iewInclusive	The selection is a closed polygon.
iewExclusive	The selection includes only points that match the initial pixel, like a flood fill
iewGlobal	The selection includes all points that match the initial pixel, looking for all pixels in the image.

DECLARATION

```
property MagicWandTolerance: integer;
```

DESCRIPTION

MagicWandTolerance specifies the color difference of original point and region point (default 15).

Unit ImageEnView

TImageEnView

DECLARATION

```
property MouseInteract: TIEMouseInteract;
```

DESCRIPTION

MouseInteract determines which mouse activities TImageEnView will handle automatically.

DECLARATION

```
type TIEMouseInteract = set of
TIEMouseInteractItems;
```

User Actions

Unit ImageEnView
TImageEnView

DECLARATION

TIEMouseInteractItems = (miZoom, miScroll, miSelect, miSelectPolygon, miSelectCircle, miSelectZoom, miSelectMagicWand, miSelectLasso, miMoveLayers, miResizeLayers, miRotateLayers, miMovingScroll);

DESCRIPTION

Value	Description
miZoom	left click zoom-in the image, otherwise right click zoom-out.
miScroll	user can navigate the image with a left click and mouse movement.
miSelect	User can select a rectangular area. Click and move mouse to select the rectangle. miSelect excludes miScroll and miSelectXXX. SHIFT key allows selection of multiple regions. Simultaneously pressing ALT causes the selection to maintain its aspect ratio.
miSelectZoom	User can select an area, which will be zoomed-in. Click and move mouse to zoom the rectangle.
miSelectPolygon	User can select a polygonal area. Press and release left mouse button for each vertex of polygon, otherwise press left button and move mouse to specify continuous irregular lines. The SHIFT key selects multiple regions.

Value	Description
miSelectCircle	User can select a circular (ellipse) area. Press left button and move mouse (press ALT to select a circle). SHIFT key selects multiple regions.
miSelectMagicWand	User can select an irregular region with similar colors. Press and release the left button over a point. SHIFT key selects multiple regions.
miSelectLasso	User can select a polygonal area. Press the left button and move mouse to specify continuous irregular region.
miMoveLayers	User can move layers. LayersSync must be False.
miResizeLayers	User can resize layers using grips. LayersSync must be False.
miRotateLayers	User can rotate layers using grips. LayersSync must be False.
miMovingScroll	User can smooth scroll (pan) image just moving mouse pointer.

Unit ImageEnView
TImageEnView

DECLARATION

```
property MouseScrollRate: double;
```

DESCRIPTION

If MouseInteract contains miScroll then MouseScrollRate specifies the relation between mouse movement and image scroll. The default is 1, which means that one mouse point is one pixel (at 100%).

Example

```
// set double rate scroll
ImageEnView1.MouseScrollRate:=2;
ImageEnView1.MouseInteract:=[miScroll];
```

Unit ImageEnView
TImageEnView

DECLARATION

```
property MouseWheelParams: TIEMouseWheelParams;
```

DESCRIPTION

The MouseWheelParams property allows application to customize the mouse wheel's behavior.

Example

```
// mouse wheel will scroll image of 15% of
// component height
ImageEnView1.MouseWheelParams.Action := iemwVScroll;
ImageEnView1.MouseWheelParams.Variation := iemwPercentage;
ImageEnView1.MouseWheelParams.value := 15;

// mouse wheel will scroll image of 1 pixel
ImageEnView1.MouseWheelParams.Action := iemwVScroll;
ImageEnView1.MouseWheelParams.Variation := iemwAbsolute;
ImageEnView1.MouseWheelParams.value := 1;

// prevent MouseWheel zooming
ImageEnView1.MouseWheelParams.Action:=iemwNone;
```

Unit ImageEnView

TImageEnView

DESCRIPTION

The TIEMouseWheelParams allows applications to customize the mouse wheel's behavior.

Properties

DECLARATION

```
InvertDirection: Boolean;
```

DESCRIPTION

If InvertDirection is true inverts wheel direction (default False).

DECLARATION

Action: TIEMouseWheelParamsAction;

DESCRIPTION

Action specifies the task to perform on mouse wheel events. Specify iemwNone for no operation, iemwVScroll for vertical image scroll or iemwZoom for zoom-in/out (default iemwZoom).

Unit ImageEnView
TImageEnView

DECLARATION

Variation: TIEMouseWheelParamsVariation;

DESCRIPTION

Variation specifies how much scrolling or zooming occurs in response to mouse wheel rotation. If Variation is iemwAbsolute, Value contains the absolute value to add or subtract from the current value. If Variation is iemwPercentage, Value contains the percentage of variation from the current value (default is iemwPercentage).

DECLARATION

Value: integer;

DESCRIPTION

Value is the percentage of variation (default 8).

DECLARATION

ZoomPosition: TIEMouseWheelParamsZoomPosition;

DESCRIPTION

If Action is iemwZoom, ZoomPosition specifies where the zoom acts. The default is the center of the control, otherwise (iemwMouse) zooms from the mouse's position.

User Actions

Unit ImageEnView

TImageEnView

DECLARATION

```
property ZoomSelectionAspectRatio: Boolean;
```

DESCRIPTION

This property is active when MouseInteract contains miSelectZoom. If true (default) the selected rectangle is adjusted to maintain aspect ratio. Otherwise (false) the image losses aspect ratio (ZoomX<>ZoomY and Zoom value is invalid) making it stretched inside the component.

Selections

DECLARATION

```
procedure AddSelBreak;
```

DESCRIPTION

AddSelBreak breaks current selection to begin a new selection.

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure AddSelPoint(x, y: integer);
```

DESCRIPTION

AddSelPoint adds a point to the current polygonal selection. If SelectionBase is iesbClientArea (default), all coordinates depend upon actual zoom and scrolling. Otherwise, if SelectionBase is iesbBitmap all coordinates refer to bitmap pixels. Use EndSelect to terminate selection by code.

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure AssignSelTo(Dest: TPersistent);
```

DESCRIPTION

Assigns the selected area to the Dest object (can be a TImageEnView, TBitmap or TImage).

Example

```
// Create a triangular selection
ImageEnView1.DeSelect;
ImageEnView1.AddSelPoint(100,100);
ImageEnView1.AddSelPoint(200,100);
ImageEnView1.AddSelPoint(150,50);
ImageEnView1.EndSelect;

// assign selection to ImageEnView2
ImageEnView1.AssignSelTo( ImageEnView2 );

// assign selection to mybitmap (TBitmap)
ImageEnView1.AssignSelTo( mybitmap );
```

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure CopySelectionToBitmap(DestBitmap:
TBitmap);
```

DESCRIPTION

CopySelectionToBitmap copies the current selection to the specified bitmap.

Example

```
ImageEnView1.CopySelectionToBitmap(
ImageEnView2.Bitmap );
```

DECLARATION

```
procedure CopySelectionToIEBitmap(DestBitmap:
TIEBitmap; FillBackground: Boolean);
```

DESCRIPTION

CopySelectionToIEBitmap copies the current selection to the specified bitmap. If FillBackground is true the background is filled using Background color, otherwise it is filled with image self.

Example

ImageEnView1.CopySelectionToIEBitmap(ImageEnView2.IEBitmap);

Selections

Unit ImageEnView
TImageEnView

DECLARATION

```
property DelayDisplaySelection: Boolean
```

DESCRIPTION

If DelayDisplaySelection is true, the selection is displayed with a delay. This allows you to quickly navigate the image (zoom and scroll) and only after you have finished the navigation the selection is displayed.

DECLARATION

```
procedure DeSelect;
```

DESCRIPTION

This method removes current selection. After DeSelect is called, the selected property will be false.

Unit ImageEnView
TImageEnView

DECLARATION

```
function DiscardSavedSelection: Boolean;
```

DESCRIPTION

DiscardSavedSelection removes the last saved selection from the stack (saved using SaveSelection). You can use this method if you don't want restore the saved selection.

DECLARATION

```
procedure EndSelect;
```

DESCRIPTION

EndSelect terminates a selection specified by code using AddSelPoint and AddSelBreak methods.

Example

```
ImageEnView1.AddSelPoint(0,0);
ImageEnView1.AddSelPoint(100,100);
ImageEnView1.AddSelPoint(50,50);
ImageEnView1.EndSelect;
```

Selections

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure GetSelectionGripStyle(var GripColor1: TColor; var GripColor2: TColor; var GripBrushStyle: TBrushStyle; var GripSize: integer; var ExtendedGrips: Boolean; var Shape: TIEGripShape);
```

DESCRIPTION

GetSelectionGripStyle returns a property which determines the appearance of selection grips.

Parameter	Description
GripColor1	grip border color (default clBlack)
GripColor2	grip brush color (default clWhite)
GripBrushStyle	brush style (default bsSolid)
GripSize	size in pixels of the grip (default 5)
ExtendedGrips	if true enables grips on border with 8 resizing grips (default false)
Shape	specifies the grip shape

Selections

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure InvertSelection;
```

DESCRIPTION

InvertSelection inverts current selection.

Example

```
ImageEnView1.Select(10,10,100,100, iespReplace);
// select box 10,10,100,100
ImageEnView1.InvertSelection; // select all
excluding the box 10,10,100,100
```

DECLARATION

```
function IsPointInsideSelection(x, y: integer):
Boolean;
```

DESCRIPTION

IsPointInsideSelection returns true if the point specified with x, y is inside the current selection. If SelectionBase is iesbClientArea (default) all coordinates depend on actual zoom and scrolling.

Unit ImageEnView
TImageEnView

DECLARATION

```
function LoadSelectionFromFile(const FileName:
string; Options: TIERSOptions = iersMoveToAdapt):
Boolean;
```

DESCRIPTION

LoadSelectionFromFile loads a selection from a file (saved using SaveSelectionToStream or SaveSelectionToFile). LoadSelectionFromFile returns false if it fails. Options specify how the selection is adapted when it comes from an image with different size or position. If you want share the same selection among layers you have to use iersSyncLayers.

Example

```
ImageEnView1.Select(10,10,100,100);
ImageEnView1.SaveSelectionToFile('selection1');
..
sel1.Position:=0;
ImageEnView1.LoadSelectionFromFile('selection1');
// this is like Select(10,10,100,100)
```

Unit ImageEnView
TImageEnView

DECLARATION

```
function LoadSelectionFromStream(Stream: TStream;
Options: TIERSOptions = iersMoveToAdapt):
Boolean;;
```

DESCRIPTION

LoadSelectionFromStream loads a selection from a stream (saved using SaveSelectionToStream or SaveSelectionToFile). LoadSelectionFromStream returns false if it fails. An option specifies how the selection is adapted when it comes from an image with different size or position.

If you want share the same selection among layers you have to use iersSyncLayers.

Example

```
ImageEnView1.Select(10,10,100,100);
ImageEnView1.SaveSelectionToStream(sel1);
..
sel1.Position:=0;
ImageEnView1.LoadSelectionFromStream(sel1); //
this is like Select(10,10,100,100)
```

Selections

Unit ImageEnView
TImageEnView

DECLARATION

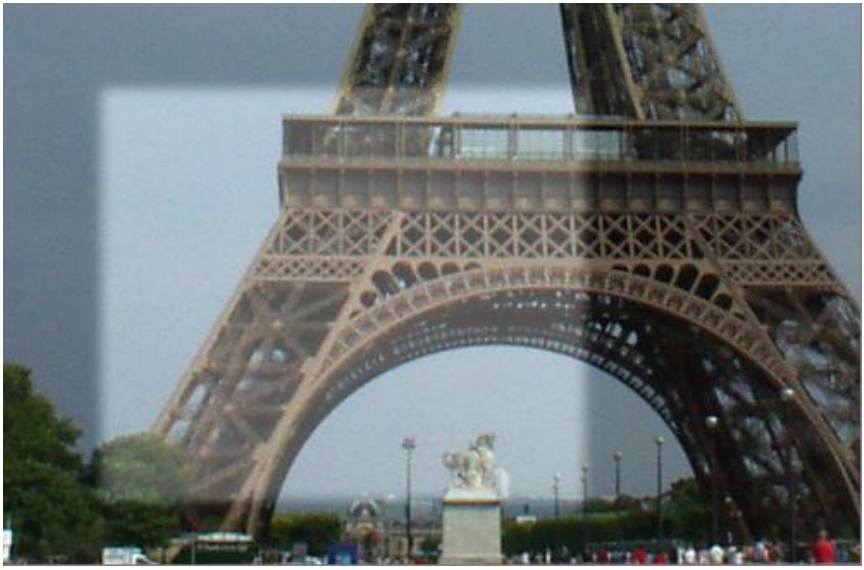
```
procedure MakeSelectionFeather(Radius: double);
```

DESCRIPTION

MakeSelectionFeather modifies the current selection by applying a blur effect to make borders soft. This property works only if SelectionMaskDepth is 8. Radius specifies the feathering intensity (blur intensity).

Example

```
ImageEnView1.SelectionMaskDepth := 8;
ImageEnView1.Select( 10,10, 100,100 );
ImageEnView1.MakeSelectionFeather( 4 );
```



Selections

Unit ImageEnView
TImageEnView

DECLARATION

```
function MergeSelectionFromFile(const
FileName:string):Boolean;
```

DESCRIPTION

MergeSelectionFromFile loads a selection from a file (saved using SaveSelectionToStream or SaveSelectionToFile) merging it with the currently one. MergeSelectionFromFile returns false if it fails.

Example

```
ImageEnView1.Select(10,10,100,100);
ImageEnView1.SaveSelectionToFile('selection1');
..
sel1.Position:=0;
ImageEnView1.MergeSelectionFromFile('selection1')
; // this is like Select(10,10,100,100, iespAdd)
```

Unit ImageEnView
TImageEnView

DECLARATION

```
function MergeSelectionFromStream(Stream:
TStream): Boolean;
```

DESCRIPTION

MergeSelectionFromStream loads a selection from a stream (saved using SaveSelectionToStream or SaveSelectionToFile) merging it with the currently one. MergeSelectionFromStream returns false if it fails.

Example

```
ImageEnView1.Select(10,10,100,100);
ImageEnView1.SaveSelectionToStream(sel1);
..
sel1.Position:=0;
ImageEnView1.MergeSelectionFromStream(sel1); //
this is like Select(10,10,100,100, iespAdd)
```

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure MoveSelection(OffsetX, OffsetY:
integer);
```

DESCRIPTION

MoveSelection moves the current selection by the specified horizontal and vertical offsets.

Example

```
ImageEnView1.MoveSelection( -5, -5)'
```

DECLARATION

```
property PolySel[idx: integer]:TPoint;
```

DESCRIPTION

PolySel returns the idx point of the current polygonal selection. The point is specified in bitmap coordinates. See also PolySelCount.

Read only

Unit ImageEnView

TImageEnView

DECLARATION

```
property PolySelCount: integer;
```

DESCRIPTION

Returns the points number of the current polygonal selection (size of PolySel array). See also PolySel.

Read only

Unit ImageEnView
TImageEnView

DECLARATION

```
property PolySelPoints: PPointArray;
```

DESCRIPTION

PolySelPoints returns the current selection as a pointer to an array of TPoint.

Example

```
// Copies current selection to ImageEnView2. Then
// enhance contrast of ImageEnView2 and copy back
// to ImageEnView1 (in some selection).
ImageEnView2.CopyFromPolygon(ImageEnView1.Bitmap,ImageEnView
2.PolySelPoints^,ImageEnView2.PolySelCount,Point(0,0));
ImageEnView2.Proc.Contrast(30);
ImageEnView2.CopyToPolygon(ImageEnView1.Bitmap,ImageEnView1.P
olySelPoints^,ImageEnView1.PolySelCount,Point(0,0));
ImageEnView1.Update;
```

Unit ImageEnView

TImageEnView

DECLARATION

```
PPointArray = ^TPointArray;
```

Selections

Unit ImageEnView
TImageEnView

DECLARATION

```
function RestoreSelection(Remove: Boolean = true;
Options: TIERSOptions = iersMoveToAdapt):
Boolean;
```

DESCRIPTION

RestoreSelection restores a saved selection from the selections stack. RestoreSelection returns false when there aren't selections saved in the stack.

An option specifies how the selection is adapted when it comes from an image with different size or position. If you want share the same selection among layers you have to use iersSyncLayers. Remove allows you to remove restored selection from selections stack.

Example

```
ImageEnView1.Select(10,10,100,100);
ImageEnView1.SaveSelection;
ImageEnView1.Select(200,200,150,150);
ImageEnView1.SaveSelection;
ImageEnView1.Deselect;

ImageEnView1.RestoreSelection; // reload
200,200,150,150
..
ImageEnVIew1.RestoreSelection; // reload
10,10,100,100
```

Selections

Unit ImageEnView

TImageEnView

DECLARATION

```
TIERSOptions = (iersNone, iersMoveToAdapt,
iersSyncLayers);
```

DESCRIPTION

Value	Description
iersNone	no adaption
iersMoveToAdapt	move the selection to fit inside the new size
iersSyncLayers	used to maintain the selection in the same position among layers

Selections

Unit ImageEnView

TImageEnView

DECLARATION

```
property SavedSelectionsCount: integer
```

DESCRIPTION

SavedSelectionsCount returns the number of saved selections.

DECLARATION

```
procedure SaveSelection;
```

DESCRIPTION

SaveSelection pushes the current selection in a stack.

Example

```
ImageEnView1.Select(10,10,100,100);
ImageEnView1.SaveSelection;
ImageEnView1.Select(200,200,150,150);
ImageEnView1.SaveSelection;
ImageEnView1.Deselect;

ImageEnView1.RestoreSelection; // reload
200,200,150,150
..
ImageEnView1.RestoreSelection; // reload
10,10,100,100
```

Selections

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure SaveSelectionToFile(const
FileName:string);
```

DESCRIPTION

SaveSelectionToFile saves the current selection to the specified file.

Example

```
ImageEnView1.Select(10,10,100,100);
ImageEnView1.SaveSelectionToFile('selection1');
..
sel1.Position:=0;
ImageEnView1.LoadSelectionFromFile('selection1');
// this is like Select(10,10,100,100)
```

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure SaveSelectionToStream(Stream: TStream);
```

DESCRIPTION

SaveSelectionToStream saves the current selection to the specified stream.

Example

```
ImageEnView1.Select(10,10,100,100);
ImageEnView1.SaveSelectionToStream(sel1);
..
sel1.Position:=0;
ImageEnView1.LoadSelectionFromStream(sel1); //
this is like Select(10,10,100,100)
```

DECLARATION

property SelColor1: TColor

DESCRIPTION

SelColor1 sets first color of the animated selection polygon.

Unit ImageEnView
TImageEnView

DECLARATION

```
property SelColor2: TColor
```

DESCRIPTION

SelColor2 sets the second color of the animated selection polygon.

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure Select(x1, y1, x2, y2: integer; Op:
TIESelOp = iespReplace); virtual;
```

DESCRIPTION

This method selects a portion of the image. The selected area will be outlined with an animated rectangle. If SelectionBase is iesbClientArea (default) all coordinates depend on actual zoom and scrolling. If SelectionBase is iesbBitmap, all coordinates refer to bitmap pixels.

Parameter	Description
x1	Top-left horizontal position.
y1	Top-left vertical position.
x2	Bottom-right horizontal position. Last column not selected.
y2	Bottom-right vertical position. Last row not selected.
Selection operation	(add or replace selection).
Op	If Op is iespReplace, Select replaces all previous selections. If Op is iespAdd, Select adds a new selection.

x2, y2 represent the right/bottom margin (not selected). To select an area of 1x1 pixels we need to select using 0,0,1,1 (the column 1 and the row 1 aren't selected).

See also

SelectEllipse
SelectMagicWand
SelectRoundRect
InvertSelection
DeSelect

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure SelectColors(StartColor, FinalColor:
TRGB; Op: TIESelOp);
procedure SelectColors(Color: TRGB; Op:
TIESelOp);
procedure SelectColors(ColorIndex: integer; Op:
TIESelOp);
```

DESCRIPTION

SelectColors selects all colors inside the range StartColor up to FinalColor, or specific color or a color index. If Op is iespReplace the region replaces the existing selection, otherwise if Op is iespAdd, the region is appended to the existing selection.

The third overload accepts ColorIndex, which is the index of a color in image colormap: the bitmap pixelformat must be ie8p (palette).

Example

```
// select from 200,200,200 to 255,255,255
ImageEnView1.SelectColors(CreateRGB(200,200,200),
CreateRGB(255,255,255));

// select only color 255,0,0 (pure red)
ImageEnView1.SelectColors(CreateRGB(255,0,0));

// select color index 5 of a paletted bitmap
// (ie8p)
ImageEnView1.SelectColors(5);
```

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure SelectCustom;
```

DESCRIPTION

Call SelectCustom after you have specified a pixmap of selected pixels of the image. To select a pixmap use SelectionMask methods like SetPixel.

Example

```
// only pixels at 10,10 and 15,15 are selected
ImageEnView1.SelectionMask.SetPixel(10,10, 1);
ImageEnView1.SelectionMask.SetPixel(15,15, 1);
ImageEnView1.SelectCustom;
```

DECLARATION

```
property Selected: Boolean;
```

DESCRIPTION

Selected is true if any portion of the image is selected. Selected is true after you have called the Select method. Selected is False after you have called DeSelect method. If Selected is true, the properties SelX1, SelY1, SelX2, SelY2 are valid.

Read-only

Selections

Unit ImageEnView
TImageEnView

DECLARATION

```
property SelectedRect: TIERectangle;
```

DESCRIPTION

SelectedRect returns current selected rectangle. If no selected exists returns the whole image rectangle. The coordinate has same dimension of the Bitmap (independent from Zoom and ViewX, ViewY values). The selected area can be selected with Select method.

Read-only

DECLARATION

```
procedure SelectEllipse(CenterX, CenterY, Width,
Height: integer; Op: TIESelOp = iespReplace);
```

DESCRIPTION

SelectEllipse makes an elliptical selection, centered at CenterX and CenterY, and with size specified by Width and Height. Op specifies if you want to add a new selection or replace the current one (iespAdd or iespReplace).

Example

```
ImageEnView1.SelectEllipse( 100,100, 20,20 );
```

Unit ImageEnView
TImageEnView

DECLARATION

```
property SelectionAbsHeight: integer;
```

DESCRIPTION

SelectionAbsHeight specifies the fixed selection height, when SelectionAspectRatio is 0.

See also
SelectionAbsWidth.

Example

```
// we want a fixed selection of 100 x 100 pixels
ImageEnView1.SelectionAbsWidth := 100;
ImageEnView1.SelectionAbsHeight := 100;
ImageEnView1.SelectionAspectRatio := 0;
```

Unit ImageEnView

TImageEnView

DECLARATION

```
property SelectionAbsWidth: integer;
```

DESCRIPTION

SelectionAbsWidth specifies the fixed selection width, when SelectionAspectRatio is 0.

See also

SelectionAbsHeight.

Example

```
// we want a fixed selection of 100 x 100 pixels
ImageEnView1.SelectionAbsWidth := 100;
ImageEnView1.SelectionAbsHeight := 100;
ImageEnView1.SelectionAspectRatio := 0;
```

Unit ImageEnView
TImageEnView

DECLARATION

```
property SelectionAspectRatio: double;
```

DESCRIPTION

SelectionAspectRatio specifies the selection aspect ratio. If SelectionAspectRatio is -1, the aspect ratio is active only when the user presses the ALT key, and it is automatically calculated. If SelectionAspectRatio is 0, the size of the selection is fixed and determined by SelectionAbsWidth and SelectionAbsHeight properties. If SelectionAspectRatio is >0, ImageEn maintains the specified aspect.

Example

```
// we want standard behavior
ImageEnView1.SelectionAspectRatio:=-1;

// we want a fixed selection of 100 x 100 pixels
ImageEnView1.SelectionAbsWidth:=100;
ImageEnView1.SelectionAbsHeight:=100;
ImageEnView1.SelectionAspectRatio:=0;

// we want a fixed aspect ratio of 2
(height=2*width)
ImageEnView1.SelectionAspectRatio:=2;
```

Unit ImageEnView
TImageEnView

DECLARATION

```
property SelectionBase: TIESelectionBase;
```

DESCRIPTION

SelectionBase sets the selection coordinates base. If SelectionBase is iesbClientArea (default), all coordinates depend upon actual zoom and scrolling. Otherwise, if SelectionBase is iesbBitmap, all coordinates refer to bitmap pixels.

This property changes the coordinate specification for methods that create selections.

Example

```
// I want to select 0,0,100,100 rectangle in the //
bitmap
ImageEnView1.SelectionBase(iesbBitmap);
ImageEnView1.Select(0,0,100,100);

// I want to select 0,0,100,100 rectangle in the //
screen
ImageEnView1.SelectionBase(iesbClientArea);
ImageEnView1.Select(0,0,100,100);
```

Note: When ImageEnView1.Zoom=100, ImageEnView1.ViewX=0 and ImageEnView1.ViewY=0, the preceding examples are equivalent.

Selections

Unit ImageEnView

TImageEnView

DECLARATION

```
TIESelectionBase = (iesbClientArea,
// select coordinates over client area
iesbBitmap
// select coordinates over bitmap);
```

Selections

Unit ImageEnView
TImageEnView

DECLARATION

```
property SelectionGridSize: integer;
```

DESCRIPTION

When SelectionGridSize is > 1, a grid of SelectionGridSize size will be displayed on rectangular selections. Default is 1.

DECLARATION

```
property SelectionIntensity: integer;
```

DESCRIPTION

The SelectionIntensity property allows specifying the selection intensity and it is valid only when SelectionMaskDepth is 8 (8 bit), otherwise this must be 1. You can assign a value from 0 to 255 which will be applied to the next or current selection.

See also
SelectionMaskDepth.

Example

```
ImageEnView1.SelectionMaskDepth := 8;
ImageEnView1.SelectionIntensity := 128;
ImageEnView1.Select( 10,10,100,100 );
ImageEnView1.Proc.Negative; // the negative is
applied at 50 % (128=semi selected)
```

Unit ImageEnView
TImageEnView

DECLARATION

```
property SelectionMask: TIEMask;
```

DESCRIPTION

SelectionMask contains the current selection as a bit mask of 1 bit per pixel.

Example

```
// Excludes pixel 100,100 from selection
ImageEnView1.SelectionMask.SetPixel(100,100,0);

// To create a new selection using only
SelectionMask (this select point 50,50 and 55,55)
ImageEnView1.AddSelPoint(0,0);
ImageEnView1.EndSelect;
ImageEnView1.SelectionMask.SetPixel(50,50,1);
ImageEnView1.SelectionMask.SetPixel(55,55,1);
```

Unit ImageEnView
TImageEnView

DECLARATION

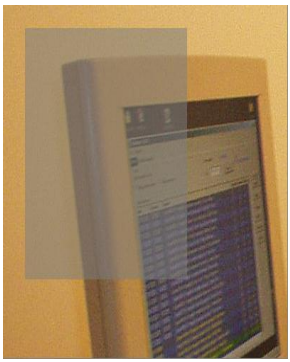
```
property SelectionMaskDepth: integer;
```

DESCRIPTION

The SelectionMaskDepth property allows specifying the selection depth in bits. The default is 1 bit, then a pixel can be “unselected”=0 or “selected” = 1. With setting 8 (8 bit), a pixel can be unselected=0, ‘semi-selected’ from 1 to 254, or fully selected =255. This allows creating soft selections or feathering selections. **Note**: SelectionOptions = iesoFilled not supported when SelectionMaskDepth=8

Example

```
ImageEnView1.SelectionMaskDepth := 8;
ImageEnView1.SelectionIntensity := 128;
ImageEnView1.Select( 10,10,100,100 );
ImageEnView1.Proc.Negative;
// the negative is applied at 50 % (128=semi
// selected)
```



Selections

Unit ImageEnView

TImageEnView

DECLARATION

```
property SelectionOptions: TIESelectionOptions;
```

DESCRIPTION

SelectionOptions controls selections behavior.

DECLARATION

```
TIESelectionOptions = set of (
iesoAnimated,
iesoSizeable,
iesoMoveable,
iesoFilled,
iesoCutBorders,
iesoMarkOuter,
iesoCanScroll,
iesoSelectTranspLayers,
iesoRightButtonSelectLayers,
iesoRightButtonTerminatePolySelect
);
```

Unit ImageEnView
TImageEnView

DESCRIPTION

Value	Description
iesoAnimated	the selection will be animated.
iesoSizeable	user can resize the selection (display grips).
iesoMoveable	user can move the selection (display grips).
iesoFilled	the selection is filled with inverted image
iesoCutBorders	user can drag the selection out of borders (cut the selection borders)
iesoMarkOuter	makes grayed the unselected area (the area out of selection)
iesoCanScroll	enables auto-scrolling of the image when selecting out of visible area
iesoSelectTranspLayers	if specified transparent areas of a layer are selectable
iesoRightButtonSelectLayers	allow to select layers with right button (other than left button)
iesoRightButtonTerminatePolySelect	right mouse button will terminates polygon selection

Selections

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure SelectMagicWand(x,y: integer; Op:
TIESelOp);
```

DESCRIPTION

SelectMagicWand selects an irregular region that has similar colors.

Parameter	Description
x	Starting horizontal coordinate.
y	Starting vertical coordinate.
Op	If Op is iespReplace the region replaces the existing selection, otherwise if Op is iespAdd, the region is appended to the existing selection.

DECLARATION

```
TIESelOp = (iespReplace, iespAdd);
```

DESCRIPTION

iespReplace replaces all previous selections. iespAdd adds a new selection.

Selections

DESCRIPTION

```
procedure SelectRoundRect(Left, Top, Right,
Bottom, RoundWidth, RoundHeight: integer; Op:
TIESelOp = iespReplace);
```

DESCRIPTION

SelectRoundRect creates a rounded selection. Left, Top, Right, and Bottom are the rectangle coordinates. RoundWidth and RoundHeight specify round size. Op specifies if you want to add a new selection or replace the current one (iespAdd or iespReplace).

Example

```
ImageEnView1.SelectRoundRect(100,100,250,250,20,20);
```

DECLARATION

```
property SelX1: integer;
```

DESCRIPTION

SelX1 is the top-left X (column) coordinate of the selected area. The coordinate has same dimension of the Bitmap (independent from Zoom and ViewX, ViewY values). The selected area can be selected with Select method.

Read-only

Unit ImageEnView
TImageEnView

DECLARATION

```
property SelX2: integer;
```

DESCRIPTION

SelX2 is bottom-right X (column) coordinate of the selected area. The coordinate has same dimension of the Bitmap (independent from Zoom and ViewX, ViewY values). The selected area can be selected with Select method.

Read-only

DECLARATION

```
property SelY1: integer;
```

DESCRIPTION

SelY1 is the top-left Y (row) coordinate of the selected area. The coordinate has same dimension of the Bitmap (independent from Zoom and ViewX, ViewY values). The selected area can be selected with Select method.

Read-only

Unit ImageEnView
TImageEnView

DECLARATION

```
property SelY2: integer;
```

DESCRIPTION

SelY2 is the bottom-right Y (row) coordinate of the selected area. The coordinate has same dimension of the Bitmap (independent from Zoom and ViewX, ViewY values). The selected area can be selected with Select method.

Read-only

Selections

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure SetSelectionGripStyle(GripColor1,
GripColor2:TColor; GripBrushStyle: TBrushStyle;
GripSize: integer; ExtendedGrips: Boolean;
Boolean; Shape: TIEGripShape);
```

DESCRIPTION

SetSelectionGripStyle determines the appearance of selection grips.

Parameter	Description
GripColor1	grip border color (default clBlack)
GripColor2	grip brush color (default clWhite)
GripBrushStyle	brush style (default bsSolid)
GripSize	size in pixels of the grip (default 5)
ExtendedGrips	if true enables grips on border with 8 resizing grips (default true)
Shape	specifies the grip shape

Use GetSelectionGripStyle to know current values.

Example

```
ImageEnView1.SetSelectionGripStyle(clWhite,
clWhite, bsSolid, 5, true, iegsCircle);
Selections Links
```

DECLARATION

```
TIEGripShape = (iegsBox, iegsCircle);
```

DECLARATION

```
procedure SetSelectionMarkOuterStyle(Alpha:
integer; Color: TColor);
```

DESCRIPTION

When SelectionOptions is iesoMarkOuter, this method specifies how outer area will be drawn. If Alpha = -1 (default) a grid of grays is displayed. If Alpha>=0 and <=255 a solid color is painted.

Example

```
// out of selection areas will be marked with red
//(with 150 transparency)
ImageEnView1.SelectionOptions:=ImageEnView1.Selec
tionOptions+[iesoMarkOuter];
ImageEnView1.SetSelectionMarkOuterStyle(150,
clRed);
```

Selections

Unit ImageEnView

TImageEnView

DECLARATION

```
property VisibleSelection: Boolean;
```

DESCRIPTION

VisibleSelection shows (True) or hides (False) the current selection. When selection is hidden, this property is still valid.

Run Time

Unit ImageEnView

TImageEnView

DESCRIPTION

TIEMask contains a selection, which is a map of selected and unselected pixels.

A selection map can have a depth of 1 bit or 8 bit.
For a map of 1 bit, 0 is a non-selected pixel, while 1 is selected.
For a map of 8 bit, 0 is non-selected pixel, >0 is “semi” selected pixel up to 255 that means fully selected pixel.

TImageEnView component uses this class to store current selection in SelectionMask property.

TIEMask

Methods and Properties

DECLARATION

```
procedure AllocateBits(width, height: integer;
bitsperpixel: integer);
```

DESCRIPTION

AllocateBits allocates memory for the map.
Memory is zero filled.
Bitsperpixel can be 1 or 8.

IEMask

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure Assign(Source: TIEMask);
```

DESCRIPTION

Assign assigns Source mask.

DECLARATION

```
property Bits: pbyte;
```

DESCRIPTION

Bits contain the raw buffer of the selection mask.

DECLARATION

```
property BitsPerPixel: integer;
```

DESCRIPTION

BitsPerPixel is the bits per pixels of the mask. ImageEn supports only 1 bit selection masks (1 selected, 0 not selected).

(**ReadOnly**)

IEMask

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure CombineWithAlpha(SourceAlpha:
TIEBitmap; ox, oy: integer;
SynchronizeBoundingRect: Boolean);
```

DESCRIPTION

Only for internal use.

DECLARATION

```
procedure CopyBitmap(Dest, Source: TBitmap;
dstonlymask, srconlymask: Boolean);
```

DESCRIPTION

For internal use only.

IEMask

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure CopyIEBitmap(Dest, Source: TIEBitmap;
dstonlymask, srconlymask: Boolean;
UseAlphaChannel: Boolean);
```

DESCRIPTION

For internal use only.

DECLARATION

```
procedure CopyIEBitmapAlpha(Dest, Source:
TIEBitmap; dstonlymask, srconlymask: Boolean);
```

DESCRIPTION

For internal use only.

IEMask

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure DrawOuter(Dest: TBitmap; xDst, yDst,
dxDst, dyDst: integer; xMask, yMask, dxMask,
dyMask: integer; AlphaBlend: integer; Color:
TColor);
```

DESCRIPTION

For internal use only.

DECLARATION

```
procedure DrawOutline(Dest: TCanvas; xDst, yDst,
dxDst, dyDst: integer; xMask, yMask, dxMask,
dyMask: integer; AniCounter: integer; Color1,
Color2: TColor; actualRect: PRect = nil);
```

DESCRIPTION

For internal use only.

IEMask

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure DrawPolygon(Alpha: integer; SelPoly:
PPointArray; SelPolyCount: integer);
```

DESCRIPTION

DrawPolygon draws the specified polygon in the mask, using Alpha value for all pixels.

DECLARATION

```
procedure Empty;
```

DESCRIPTION

Empty fills the entire mask with zeros.

IEMask

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure Fill(Alpha: integer);
```

DESCRIPTION

Fill fills the entire mask using Alpha value.

DECLARATION

```
procedure FreeBits;
```

DESCRIPTION

FreeBits free the allocated mask.

DECLARATION

```
property Full: Boolean;
```

DESCRIPTION

Full is true when the mask contains all 1 values (that is the image has all pixels selected).

IEMask

Unit ImageEnView

TImageEnView

DECLARATION

```
function GetPixel(x, y: integer): integer;
```

DESCRIPTION

GetPixel returns the value of the pixel at x,y coordinates.

DECLARATION

```
property Height: integer;
```

DESCRIPTION

Height is the height of the selection mask. It must be equal to the image height. (**ReadOnly**)

DECLARATION

```
procedure Intersect(x1, y1, x2, y2: integer);
```

DESCRIPTION

For internal use only.

IEMask

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure InvertCanvas(Dest: TCanvas; xDst, yDst,
dxDst, dyDst: integer; xMask, yMask, dxMask,
dyMask: integer);
```

DESCRIPTION

For internal use only.

DECLARATION

```
function IsEmpty: Boolean;
```

DESCRIPTION

IsEmpty returns true is the mask contains all zeros.

DECLARATION

```
function IsPointInside(x,y:integer):Boolean;
```

DESCRIPTION

IsPointInside returns true if the x,y pixel is not 0.

IEMask

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure Negative(MaxVal: integer);
```

DESCRIPTION

For internal use only.

DECLARATION

```
procedure Resize(NewWidth, NewHeight: integer);
```

DESCRIPTION

Resize resizes this selection map.

DECLARATION

```
property Rowlen: integer;
```

DESCRIPTION

RowLen is the length of a row in bytes. (**ReadOnly**)

Unit ImageEnView

TImageEnView

DECLARATION

```
property ScanLine[row: integer]: pointer;
```

DESCRIPTION

Like TBitmap.Scanline property. It allows to getting and setting the mask by hand.

DECLARATION

```
procedure SetPixel(x, y: integer; Alpha:
integer);
```

DESCRIPTION

SetPixel sets a single pixel selection at x,y coordinates. A pixel with mask 1 or >0 is selected.

DECLARATION

```
procedure SyncFull;
```

DESCRIPTION

SyncFull sets Full to true if all values are 255.

IEMask

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure SyncRect;
```

DESCRIPTION

SyncRect adjusts X1, Y1, X2 and Y2 to enclose the bounding box of the selected pixels.

DECLARATION

```
procedure TranslateBitmap(var ox, oy: integer;
CutSel: Boolean);
```

DESCRIPTION

TranslateBitmap translates the mask of ox, oy pixels. CutSel is true if the mask can go out of mask margins.

DECLARATION

```
property Width: integer;
```

DESCRIPTION

Width is the width of the selection mask. It must be equal to the image width. (**ReadOnly**)

IEMask

Unit ImageEnView
TImageEnView

DECLARATION

```
property X1: integer;
```

DESCRIPTION

X1 is the left-up side of the non-empty selection (an empty mask has 1).

DECLARATION

```
property X2: integer;
```

DESCRIPTION

X2 is the right-bottom side of the non-empty selection.

DECLARATION

```
property Y1: integer;
```

DESCRIPTION

Y1 is the left-up side of the non-empty selection (an empty mask has 1).

Unit ImageEnView
TImageEnView

DECLARATION

```
property Y2: integer;
```

DESCRIPTION

Y2 is the right-bottom side of the non-empty selection.

IEMask

Transitions

DECLARATION

```
procedure AbortTransition;
```

DESCRIPTION

This method aborts current transition started with RunTransition.

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure PrepairTransition;
```

DESCRIPTION

PrepareTransition copies the currently displayed image to an internal buffer. This allows you to change current image (load, update, etc.) before starting the transition with RunTransition. An image must always be present.

Example

```
// In this example ImageEnView1 already contains
// a image
ImageEnView1.PrepareTransition;
// this prepare the transition
ImageEnView1.IO.LoadFromFile('image2.jpg');
// load the image, but it is not displayed until
// next paint
ImageEnView1.RunTransition(iettRandompoints,500);
// this executes the transition
```

Example

```
// An existing image must be always present. This
// fills with a black image and runs
//iettCrossDissolve. iettCrossDissolve fades from
// black:
ImageEnView1.IEBitmap.Allocate(ImageEnView1.Width
, ImageEnView1.Height);
ImageEnView1.IEBitmap.Fill(clBlack);
ImageEnView1.PrepareTransition();
ImageEnView1.IO.LoadFromFile('yourimage.jpg');
ImageEnView1.RunTransition(iettCrossDissolve,
2000);
```

Example

```
// An existing image must be always present. This
// fills with a white image and runs
// iettAngledTwistIn
ImageEnView1.IEBitmap.Allocate (
   ImageEnView1.Width, ImageEnView1.Height );
ImageEnView1.IEBitmap.Fill ( clWhite );
ImageEnView1.Proc.CastAlpha ( 0, 0, 0, 1 );
ImageEnView1.PrepareTransition ( );
ImageEnView1.IEBitmap.Assign ( FIEImageList.Image
[ Item.Index ] );
ImageEnView1.Update;
ImageEnView1.RunTransition ( iettAngledTwistIn,
250 );
```

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure RunTransition(Effect:
TIETransitionType; duration: integer);
```

DESCRIPTION

RunTransition starts the transition using Effect and Duration parameters. Effect specifies the effect. Duration specifies the duration of the transition in milliseconds.

Example

```
// we suppose that ImageEnView1 already contains
// an image
ImageEnView1.PrepareTransition;
// this prepare the transition
ImageEnView1.IO.LoadFromFile('image2.jpg');
// load the image, but it is not displayed until
// next paint
ImageEnView1.RunTransition(iettRandompoints,500);
// this executes the transition
```

Example

```
    //An existing image must be always present.
//This fills with a black image and runs
//iettCrossDissolve. iettCrossDissolve fades from
// black:
ImageEnView1.IEBitmap.Allocate(ImageEnView1.Width,
ImageEnView1.Height);
ImageEnView1.IEBitmap.Fill(clBlack);
ImageEnView1.PrepareTransition();
ImageEnView1.IO.LoadFromFile('yourimage.jpg');
ImageEnView1.RunTransition(iettCrossDissolve,
2000);
```

Unit ImageEnView
TImageEnView

DECLARATION

```
property TransitionEndRect: TRect;
```

DESCRIPTION

TransitionEndRect specifies the ending rectangle of iettPanZoom transition. Coordinates are in bitmap coordinates. To specify the starting rectangle set TransitionStartRect.

DECLARATION

```
property TransitionRectMaintainAspectRatio:
Boolean;
```

DESCRIPTION

This property is used when you set a value for TransitionStartRect property. If TransitionRectMaintainAspectRatio is true (default), the transition rectangle is adjusted to maintain aspect ratio.

DECLARATION

```
property TransitionRunning: Boolean;
```

DESCRIPTION

TransitionRunning is true whenever a transition is running.

Unit ImageEnView
TImageEnView

DECLARATION

```
property TransitionStartRect: TRect;
```

DESCRIPTION

TransitionStartRect specifies the starting rectangle of iettPanZoom transition. Coordinates are specified in bitmap points. To specify the ending rectangle set TransitionEndRect.

DECLARATION

```
property TransitionTiming: TIETransitionTiming;
```

DESCRIPTION

TransitionTiming allows you to select how the transition progresses.

DECLARATION

```
TIETransitionTiming=(iettLinear, iettExponential,
iettLogarithmic);
```

Unit ImageEnView

TImageEnView

Layers

DECLARATION

```
property CurrentLayer: TIELayer;
```

DESCRIPTION

Gets access to the current layer info. This is the same as: ImageEnView.Layers[ImageEnView.LayersCurrent].

Look at Layers property for more info.

Example

```
ImageEnView1.LayersAdd;
ImageEnView1.CurrentLayer.Transparency := 200;
```

Unit ImageEnView
TImageEnView

DECLARATION

```
TIELayer
```

DESCRIPTION

This class contains a single layer of TImageEnView object. A layer has a TIEBitmap, position, size, transparency and other properties. You should create new layers using LayersAdd.

Methods and Properties

DECLARATION

```
procedure Assign(Source: TIELayer);
```

DESCRIPTION

Assign copies the content of Source layer in the current one.

DECLARATION

```
property Bitmap: TIEBitmap;
```

DESCRIPTION

Bitmap contains the image and alpha channel of the layer.

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
function ConvXScr2Bmp(x: integer): integer;
```

DESCRIPTION

ConvXScr2Bmp converts screen coordinates to bitmap coordinates, taking care the layer position and size.

DECLARATION

```
function ConvYScr2Bmp(y: integer): integer;
```

DESCRIPTION

ConvYScr2Bmp converts screen coordinates to bitmap coordinates, taking care the layer position and size.

DECLARATION

```
function ConvXBmp2Scr(x: integer): integer;
```

DESCRIPTION

Converts bitmap coordinates to screen coordinates, taking care the layer position and size.

Layers

Unit ImageEnView
TImageEnView

Declaration

```
function ConvYBmp2Scr(y: integer): integer;
```

Description

Converts bitmap coordinates to screen coordinates, taking care the layer position and size.

Declaration

```
property Visible: Boolean;
```

Description

Set to True to make the layer visible, otherwise set to False to hide it.

Declaration

```
property VisibleBox: Boolean;
```

Description

If VisibleBox is true, a box around the layer will be displayed.

Unit ImageEnView
TImageEnView

DECLARATION

```
property Transparency: integer;
```

DESCRIPTION

Transparency allows setting the overall transparency of the layer (255=opaque..0=transparent)

DECLARATION

```
property PosX: integer;
```

DESCRIPTION

PosX specifies the relative (to the layer 0 - background image) position of the layer. Valid only when LayersSync=False.

DECLARATION

```
property PosY: integer;
```

DESCRIPTION

PosY specifies the relative (to the layer 0 - background image) position of the layer. Valid only when LayersSync=False.

Unit ImageEnView
TImageEnView

DECLARATION

```
property ClientAreaBox: TRect;
```

DESCRIPTION

This is a read-only property which will contains where ImageEn drawn the layer. The coordinates are relative to the component client area. Valid only if LayersSync=False.

DECLARATION

```
property Cropped: Boolean;
```

DESCRIPTION

If cropped is true and the layer is out of background image (layer 0) area, this layer will be cut to the background.
Valid only if LayersSync=False.

DECLARATION

```
property Locked: Boolean;
```

DESCRIPTION

If locked is false the user can move or resize the layer. Also this enables to show the layer border and resizing grips.
Valid only if LayersSync=False.

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
property Selectable:  Boolean;
```

DESCRIPTION

If selectable is true (default) the layer is selectable by user action.

DECLARATION

```
property Magnify: TIELayerMagnification
```

DESCRIPTION

Magnify sets properties when the layer is a magnification layer.

Unit ImageEnView
TImageEnView

DECLARATION

```
    TIELayerMagnification = record
Enabled: Boolean;
Rate: double;
Style:  TIEMagnifyStyle;
Source:  TIEMagnifySource;
end;
```

DESCRIPTION

Field	Description
Enabled	If true this is a magnify layer. The bitmap of the layer is filled with the background zoomed according to Rate field. Valid only if LayersSync=False. This property doesn't apply to layer 0.
Rate	Specifies the rate of the magnify layer (magnification). Allowed values greater than 1. Valid only if Enabled=True and LayersSync=False. This property doesn't apply to layer 0.
Style	Specifies the magnify layer shape. Valid only if Enabled=True and LayersSync=False. This property doesn't apply to layer 0.
Source	Specifies the magnify layer source.

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
property Operation: TIERenderOperation;
```

DESCRIPTION

Operation specifies the operation to execute between this layer and the background layer.

DECLARATION

```
property UserData: pointer;
```

DESCRIPTION

UserData contains a pointer to user data buffer.
Look also UserDataLen.

DECLARATION

```
property UserDataLen: integer;
```

DESCRIPTION

UserDataLen specifies the length of UserData buffer.
If this value is >0 then when you save the layer (LayersSaveTo) the content of UserData buffer is saved (and restored with LayersLoadFrom).

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
property ResampleFilter: TResampleFilter;
```

DESCRIPTION

ResampleFilter is actived only when UseResampleFilter is true this property specifies a resample filter to use instead of default one (ZoomFilter).

DECLARATION

```
property Rotate: double;
```

DESCRIPTION

Rotate specifies the layer rotation angle in degrees.
It is not allowed Rotation and Resize in the same time.

To allow resize after a layer rotation, it must be marked effective, actually rotate the bitmap. Vice versa, to allow rotation after a resize, the resizing must be effective, actually resizing the bitmap. To make effective layer rotation call LayersFixRotations. To make effective layer resizing call LayersFixSizes. Finally, multiple rotations can cause the bitmap add external, transparent, border. To remove this border call LayersFixBorders.

Layers

When rotations are handled by MouseInteract (setting miRotateLayers), you don't need to call LayersFixRotations, LayersFixSizes and LayersFixBorders.Layers Links.

Demo
imageprocessing1\layers
imageprocessing1\RotateLayers

DECLARATION

```
property RotateCenterX: double;
```

DESCRIPTION

RotateCenterX specifies the horizontal rotation center, in percentage of layer size. 0.5 means center (default), while 0 is left side and 1 is right side. Also allowed <0 and >1 values. This value (and RotateCenterY) is modified when user moves the central grip, when layers rotation is active.

Look also
Rotate.

Unit ImageEnView

TImageEnView

DECLARATION

```
property RotateCenterY: double;
```

DESCRIPTION

RotateCenterY specifies the vertical rotation center, in percentage of layer size. 0.5 means center (default), while 0 is top side and 1 is bottom side. Also allowed <0 and >1 values. This value (and RotateCenterX) is modified when user moves the central grip, when layers rotation is active.

Look also
Rotate.

DECLARATION

```
property UseResampleFilter: Boolean;
```

DESCRIPTION

This property active the value specified in ResampleFilter.

DECLARATION

```
property Name: AnsiString;
```

DESCRIPTION

Name contains the layer name. Only applications use this field.

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
property IsMask: Boolean;
```

DESCRIPTION

If IsMask is true this is a layer mask. A layer mask contains a gray scale image (ie8g) which is connected to the upper layer. The layer mask specifies where the upper layer is visible (255=fully visible).
A layer mask should be invisible (Visible=False).

DECLARATION

```
property DrawingInfo: TIEDrawingInfo;
```

DESCRIPTION

DrawingInfo specifies the source and destination rectangle recorded on last drawing.

DECLARATION

```
property DrawOuter: Boolean;
```

DESCRIPTION

If DrawOuter is true draws unselected layers as 'grayed'. The selected layer is drawn normally.

Unit ImageEnView

TImageEnView

DECLARATION

```
property Width: integer;
```

DESCRIPTION

Width specifies the layer size (in bitmap sizes). Valid only if LayersSync=False.

DECLARATION

```
property Height: integer;
```

DESCRIPTION

Height specifies the layer size (in bitmap sizes). Valid only if LayersSync=False.

DECLARATION

```
function FindLayerAt(x,y: integer;
SelectablesOnly: Boolean=true):integer;
```

DESCRIPTION

FindLayerAt returns the layer index under the position x, y (in client area coordinates). If SelectablesOnly is true, discards layers which aren't selectable. Returns -1 if no layer was found.

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
property Layers[idx: integer]:TIELayer;
```

DESCRIPTION

The Layers property allows setting/getting some properties related to the specified layer.

DECLARATION

```
function LayersAdd: integer;
function LayersAdd(Width: integer; Height:
integer; PixelFormat: TIEPixelFormat =
ie24RGB):integer;
function LayersAdd(Bitmap: TIEBitmap): integer;
function LayersAdd(FileName: string; PosX:
integer = 0; PosY: integer = 0): integer;
```

DESCRIPTION

LayersAdd appends a new layer in the layers list. The layer becomes the current layer. If not specified the new layer assumes the size and pixel format of current layer. LayersAdd returns the index of the added layer. When present, PosX and PosY specify destination layer position.

Layers

Example

```
ImageEnView1.IO.LoadFromFile('first.jpg'); //
load 'first' in first layer (or current layer)
ImageEnView1.LayersAdd; // append a new layer
ImageEnView1.IO.LoadFromFile('second.jpg'); //
load 'second.jpg' in the new append layer
```

DECLARATION

```
property LayersAutoUndo: Boolean;
```

DESCRIPTION

Allows auto undo for layers modifications made by user actions like moving and resizing.

DECLARATION

```
procedure LayersClear;
```

DESCRIPTION

LayersClear removes all layers. At the end this method creates a new layer, which will be the background layer because at least one layer must exist.

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure LayersCopyToAlpha(DestLayer: integer);
```

DESCRIPTION

LayersCopyToAlpha copies current layer to alpha channel of specified destination layer. This is useful to handle alpha channel has other bitmaps, applying the same image processing algorithms.

Look also LayersCreateFromAlpha method.

Example

```
ImageEnView1.IO.LoadFromFile('image.jpg');
ImageEnView1.LayersCreateFromAlpha;
ImageEnView1.Proc.BumpMapping( 300,300,150,150,0,
CreateRGB(255,255,255));
ImageEnView1.LayersCopyToAlpha(0);
ImageEnView1.LayersRemove(1);
```

DECLARATION

```
property LayersCount: integer;
```

DESCRIPTION

LayersCount returns the number of layers.

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
function LayersCreateFromAlpha: integer;
```

DESCRIPTION

LayersCreateFromAlpha creates a new layer with the content of current bitmap alpha channel. This is useful to handle alpha channel has other bitmaps, applying the same image processing algorithms.

Look also LayersCopyToAlpha method.

Example

```
ImageEnView1.IO.LoadFromFile('image.jpg');
ImageEnView1.LayersCreateFromAlpha;
ImageEnView1.Proc.BumpMapping( 300,300,150,150,0,
CreateRGB(255,255,255));
ImageEnView1.LayersCopyToAlpha(0);
ImageEnView1.LayersRemove(1);
```

DECLARATION

```
function LayersCreateFromClipboard: integer;
```

DESCRIPTION

LayersCreateFromClipboard creates a new layer with the content of clipboard. Returns index of the new layer.

Unit ImageEnView
TImageEnView

DECLARATION

```
function LayersCreateFromSelection: integer;
```

DESCRIPTION

LayersCreateFromSelection creates a new layer from current selection. This is useful, for example, to implement typical copy & paste selections. Returns index of the new layer.

Example

```
// select an ellipse
ImageEnView1.SelectEllipse(150,150,50,50);
// copy selected area and create a new layer
ImageEnView1.LayersCreateFromSelection;
// move the new layer
ImageEnView1.CurrentLayer.PosX:=100;
ImageEnView1.CurrentLayer.PosY:=100;
// paste to the background
ImageEnView1.LayersMerge(0,1);
```

Unit ImageEnView
TImageEnView

DECLARATION

```
property LayersCurrent: integer;
```

DESCRIPTION

LayersCurrent get/set the current layer. The first layer starts at 0, the last is LayersCount-1. Making a layer current changes the IEBitmap and Bitmap properties, so they point to the current layer (allowing specification of which layer is active for input/output and image processing operations).

DECLARATION

```
property LayersDrawBox: Boolean;
```

DESCRIPTION

If LayersDrawBox is true, a box is drawn around layers.

If Layers[].VisibleBox is false, no box is displayed (LayersDrawBox doesn't care).

If Layers[].VisibleBox is true and LayersDrawBox is false, a box is drawn only on the selected layer.

If Layers[].VisibleBox is true and LayersDrawBox is true, a box is drawn on all layers.

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure LayersDrawTo(Destination: TIEBitmap;
Scale: double = 1);
```

DESCRIPTION

Merges all layers and draws the result to Destination bitmap. This function should replace a sequence of LayersMerge calls. The destination bitmap will not have transparency channel.

Example

```
// draws all layers of ImageEnView1 to
// ImageEnView2
ImageEnView1.LayersDrawTo( ImageEnView2.IEBitmap );
ImageEnView2.Update;
```

DECLARATION

```
procedure LayersFixBorders(layer: integer = -1);
```

DESCRIPTION

This method removes the transparent border around the bitmap. A transparent border can be caused by multiple rotations. Layer=-1 means All layers.

Layers

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure LayersFixRotations(layer: integer = -1);
```

DESCRIPTION

Rotates the bitmap associated to each layer (or the specified layer) to the actual rotation angle. This method calls LayersFixSizes for each layer to rotate. Layer =-1 means All layers.

DECLARATION

```
procedure LayersFixSizes(layer: integer = -1);
```

DESCRIPTION

Resamples the bitmap associated to each layer (or the specified layer) to the actual layer size. Layer =-1 means All layers

DECLARATION

```
procedure LayersInsert(Position: integer);
```

DESCRIPTION

LayersInsert inserts a new layer in the layers list at the Position. The layer becomes the current layer, assumes the size of others and the pixel format of the last layer.

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
function LayersLoadFromFile(const FileName:
string): Boolean;
```

DESCRIPTION

LayersLoadFromFile loads all layers included bitmap, position, size, etc… from a file saved using LayersSaveToStream / LayersSaveToFile. It allows TImageEnView to rebuild a previous saved layers configuration. All images will be compressed to the specified format, but the file which contains all layers is an ImageEn custom file format.

LayersLoadFromFile returns true on success.

Example

```
// saves current layers configuration (compress
images as jpeg)
ImageEnView1.LayersSaveToFile('layers.lyr',
ioJPEG);

// loads saved layers
ImageEnView1.LayersLoadFromFile('layers.lyr');
```

Unit ImageEnView
TImageEnView

DECLARATION

```
function LayersLoadFromStream(Stream: TStream):
Boolean;
```

DESCRIPTION

LayersLoadFromStream loads all layers included bitmap, position, size, etc… from a file saved using LayersSaveToStream / LayersSaveToFile. It allows TImageEnView to rebuild a previous saved layers configuration. All images will be compressed to the specified format, but the file which contains all layers is an ImageEn custom file format. LayersLoadFromStream returns true on success.

Example

```
var
ims: TMemoryStream;

ims:= TMemoryStream.Create;
try
  // saves current layers configuration (compress
  // images as jpeg to memory stream)
  ImageEnView1.LayersSaveToStream (ims, ioJPEG);
  ims.Position := 0;
  // loads layer from stream
  ImageEnView1.LayersLoadFromStream(ims);
finally
  ims.Free;
end;
```

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure LayersMerge(Layer0,  Layer1:  integer;
RemoveUpperLayer: Boolean = true);
```

DESCRIPTION

LayersMerge merges Layer0 and Layer1 into one layer. The new layer has the minor index of Layer0 and Layer1. LayersMerge makes the new layer looking at Layers, Transparency and at the bitmap's alpha channels. If RemoveUpperLayer is false, the upper layer will not be removed.

Example

```
// we want to get a background image and then
// merge over it another image in
// semitransparency.
ImageEnView1.IO.LoadFromFile('background.jpg');
ImageEnView1.LayersAdd;
ImageEnView1.IO.LoadFromFile('foreground.jpg');
ImageEnView.Layers[1].Transparency:=128; // the
second layer has 128 of transparency
ImageEnView1.LayersMerge(0,1); // from now we
have only one layer
ImageEnView1.IO.SaveToFile('output.jpg');
```

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure LayersMergeAll;
```

DESCRIPTION

Call LayersMergeAll to merge all layers in one step. This method does the same job of LayersDrawTo, but replacing all layers with the merged result.

This method is fast and works with all layer options, so you should use it to merge layers before print or save to disk.

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure LayersMergeTo(Layer0, Layer1: integer;
Destination: TIEBitmap);
```

DESCRIPTION

LayersMergeTo merges Layer0 and Layer1 into a TIEBitmap object. LayersMergeTo makes the new image looking at Layers[].Transparency and at the bitmap's alpha channels. Resulting bitmap will be always 24 bit (ie24RGB).

Example

```
// we want to get a background image and then
// merge over it another image in
// semitransparency.
ImageEnView1.IO.LoadFromFile('background.jpg');
ImageEnView1.LayersAdd;
ImageEnView1.IO.LoadFromFile('foreground.jpg');
ImageEnView1.Layers[1].Transparency:=128; // the
second layer has 128 of transparency
ImageEnView1.LayersMergeTo(0,1,
ImageEnView2.IEBitmap);
ImageEnView2.IO.SaveToFile('output.jpg');
```

Layers

Unit ImageEnView

TImageEnView

DECLARATION

```
procedure LayersMove(CurIndex, NewIndex:
integer);
```

DESCRIPTION

LayersMove moves CurIndex layer to the position specified in NewIndex.

DECLARATION

```
procedure LayersRemove(idx: integer);
```

DESCRIPTION

LayersRemove removes the specified layer and frees the related bitmap. At the least one layer must be present.

Parameter	Description
idx	Index of layer to remove (0 = background/first layer)

Layers

Unit ImageEnView

TImageEnView

DECLARATION

```
property LayersResizeAspectRatio:
TIELayersResizeAspectRatio;
```

DESCRIPTION

LayersResizeAspectRatio specifies how ImageEn handles layers resizing aspect ratio. Default value is iearALTKey.

DECLARATION

```
TIELayersResizeAspectRatio = (iearDisabled,
iearALTKey, iearAlways, iearAlwaysOnCornerGrip);
```

DESCRIPTION

Value	Description
iearDisabled	No aspect ratio even pressing ALT or setting ForceALTKey).
iearALTKey	Aspect ratio only pressing ALT or setting ForceALTKey.
iearAlways	Aspect ratio always enabled on all grips.
iearAlwaysOnCornerGrip	Aspect ratio always enabled on corner grips.

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
property LayersRotateStep: integer;
```

DESCRIPTION

LayersRotateStep sets the rotate step when user rotates a layer and press SHIFT.

DECLARATION

```
property LayersRotationAntialias: Boolean;
```

DESCRIPTION

LayersRotationAntialias specifies the rotation antialiasing to use when layers rotation has finished and stabilized.

DECLARATION

```
property LayersRotationFilter: TIEAntialiasMode;
```

DESCRIPTION

LayersRotationFilter specifies the rotation filter to use when layers rotation has finished and stabilized.

Layers

Unit ImageEnView

TImageEnView

DECLARATION

```
TIEAntialiasMode = (ierFast, ierBilinear,
ierBicubic);
```

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure LayersSaveToFile(const FileName:string;
CompressionFormat: TIOFileType);
```

DESCRIPTION

LayersSaveToFile saves all layers including the bitmap, position, size, etc. to a unique file. It allows TImageEnView to rebuild a previous saved layers configuration. All images will be compressed to the specified format, but the file which contains all layers is an ImageEn custom file format.

If CompressionFormat is -1, an internal compressed format is used. This format preserves pixel format and alpha channel. If CompressionFormat is -2, an internal non-compressed format is used. This format preserves pixel format and alpha channel.

Note: CompresionFormat cannot be ioTIFF.

Example

```
// saves current layers configuration (compress
// images as jpeg)
ImageEnView1.LayersSaveToFile('layers.lyr', -1);

// loads saved layers
ImageEnView1.LayersLoadFromFile('layers.lyr');
```

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure LayersSaveToStream(Stream: TStream;
CompressionFormat: TIOFileType);
```

DESCRIPTION

LayersSaveToStream saves all layers in a unique file including the bitmap, position, size, etc.... It allows TImageEnView to rebuild a previous saved layers configuration. All images will be compressed to the specified format, but the file which contains all layers is an ImageEn custom file format.

If CompressionFormat is -1, an internal compressed format is used. This format preserves pixel format and alpha channel.

If CompressionFormat is -2, an internal non-compressed format is used. This format preserves pixel format and alpha channel.

Note: CompresionFormat cannot be ioTIFF.

See LayersLoadFromStream.

DECLARATION

```
property LayersSelectConstrains: Boolean;
```

DESCRIPTION

If true (default) selection constrains are active.
The unique layer selection constrain controlled is Selectable.

Layers

Unit ImageEnView

TImageEnView

DECLARATION

```
property LayersSync: Boolean;
```

DESCRIPTION

If LayersSync is true, all layers must have the same size.
If false (the default), layers can have different sizes and can be moved by code or user interaction (see MouseInteract).

DECLARATION

```
property MaxLayerHeight: integer;
```

DESCRIPTION

MaxLayerHeight returns the maximum height among all layers.

DECLARATION

```
property MaxLayerWidth: integer;
```

DESCRIPTION

MaxLayerWidth returns the maximum width among all layers.

Layers

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure SetLayersBoxStyle(PenStyle: TPenStyle =
psDot; PenMode: TPenMode = pmNot; PenWidth:
integer =1; PenColor: TColor = clBlack);
```

DESCRIPTION

SetLayersBoxStyle is an alternative way to change selected layer box pen style.

See also
OnDrawLayerBox

Example

```
ImageEnView1.SetLayersGripStyle(clWhite, clWhite,
bsSolid, 5, iegsCircle);
```

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure SetLayersGripStyle(GripColor1,
GripColor2: TColor; GripBrushStyle: TBrushStyle;
GripSize: integer; Shape: TIEGripShape);
```

DESCRIPTION

SetLayersGripStyle determines the appearance of layers grips.

Parameter	Description
GripColor1	grip border color
GripColor2	grip brush color
GripBrushStyle	brush style
GripSize	size in pixels of the grip
Shape	specifies the grip shape

Example

```
ImageEnView1.SetLayersGripStyle(clWhite, clWhite,
bsSolid, 5, iegsCircle);
```

Layers

Unit ImageEnView

TImageEnView

DECLARATION

```
property SoftCrop: TIESoftCropMode;
```

DESCRIPTION

SoftCrop specifies the operation to do when an upper layer is out of layer 0. When this is iesfAlphaBlend you can set the transparency using SoftCropValue. When this is iesfAdd you can specify the addend with SoftCropValue.

DECLARATION

```
TIESoftCropMode = (iesfNone, iesfAlphaBlend,
iesfGrid, iesfAdd);
```

DECLARATION

```
property SoftCropValue: integer;
```

DESCRIPTION

SoftCropValue sets the transparency of cut areas, when SoftCrop is iesfAlphaBlend. Also sets the addend value when SoftCrop is iesfAdd.

Layers

Unit ImageEnView

TImageEnView

Scroll bars

DECLARATION

```
property FlatScrollBars: Boolean;
```

DESCRIPTION

FlatScrollBars specifies whether the ImageEnView component's scroll bars are flat. ImageEn only supports flat scroll bars if the system has version 472 or later of comctl32.dll and Delphi 5/6/7 and C++Builder 5/6. This property must be set at Design Time, not at run-time.

DECLARATION

```
property HScrollBarParams: TIEScrollBarParams;
```

DESCRIPTION

The HScrollBarParams property allows an application to customize the horizontal scroll bar behavior like tracking (display refresh on mouse dragging), left/right buttons pixel scroll, pageleft/pageright pixel scroll.

ScrollBars

Unit ImageEnView
TImageEnView

DECLARATION

```
property ScrollBarsAlwaysVisible: Boolean;
```

DESCRIPTION

When the ScrollBarsAlwaysVisible property is True, the scroll bars specified in ScrollBars property will be displayed, even if this is not necessary.

DECLARATION

```
property ScrollBars: TScrollType;
```

DESCRIPTION

ScrollBars determines whether the TImageEnView control displays any scroll bars. If the component does not need scrollbars, they aren't shown.

Value	Description
ssNoneThe control has no scroll bars.	XXX
ssHorizontal	The control has a single scroll bar on the bottom edge when needed.
ssVertical	The control has a single scroll bar on the right edge when needed.
ssBoth	The control has a scroll bar on both the bottom and right edges when needed.

ScrollBars

Unit ImageEnView

TImageEnView

DECLARATION

```
property VScrollBarParams: TIEScrollBarParams;
```

DESCRIPTION

The VScrollBarParams property allows an application to customize the vertical scroll bar behavior like tracking (display refresh on mouse dragging), up/down buttons pixel scroll, pagedown/up pixel scroll.

Input/Output – *See ImageEnIO*

Image processing – *See ImageEnProc*

Animated Polygons

DECLARATION

```
procedure AnimPolygonAddPt(ap: integer; x, y: integer);
```

DESCRIPTION

Adds a new point to the polygon ap. x, y are the point coordinates (bitmap based coordinates).

Unit ImageEnView
TImageEnView

DECLARATION

```
procedure AnimPolygonClear(ap: integer);
```

DESCRIPTION

AnimPolygonClear removes all points of specified polygon.

DECLARATION

```
procedure AnimPolygonDel(ap: integer);
```

DESCRIPTION

AnimPolygonDel removes animated polygon ap (value returned from AnimPolygonNew).

Unit ImageEnView
TImageEnView

DECLARATION

```
function AnimPolygonNew(VColor1, VColor2:TColor;
VAnimated: Boolean; VSizeable: Boolean):integer;
```

DESCRIPTION

AnimPolygonNew creates a new animated polygon.

VColor1 and VColor2 specify the two colors of animation.

If VAnimated is true, the polygon is animated. If VSizeable is true, the polygon is sizeable. AnimPolygonNew returns an index of the new polygon.

Events

DECLARATION

```
property OnBeforeSelectionChange: TNotifyEvent;
```

DESCRIPTION

OnBeforeSelectionChange occurs just before the selection changes.

DECLARATION

```
property OnDrawBackBuffer: TNotifyEvent;
```

DESCRIPTION

OnDrawBackBuffer gets access to the back buffer where ImageEn will draw the image (and layers) just before the paint event. You can draw on BackBuffer handling the OnDrawBackBuffer event to paint custom graphics on it. BackBuffer is updated whenever Update is called.

Example

```
procedure Form1OnDrawBackBuffer(Sender: TObject);
begin
  with ImageEnView1.BackBuffer.Canvas do begin
    Pen.Color := clRed;
  MoveTo( 0,0 );
  LineTo( 100,100 );
  end;
end;
```

DECLARATION

```
property OnDrawBackground: TIEOnDrawBackground;
```

DESCRIPTION

OnDrawBackground occurs whenever background needs to be painted.

DECLARATION

```
property OnDrawCanvas: TIEOnDrawCanvas;
```

DESCRIPTION

OnDrawCanvas occurs whenever the component canvas is updated.

Unit ImageEnView
TImageEnView

DECLARATION

```
property OnDrawLayer: TIEDrawLayerEvent;
```

DESCRIPTION

This event occurs just after a layer is painted.

Parameter	Description
dest	The destination bitmap (usually the back buffer)
layerIndex	The layer index that we are drawing

Use Layers[].ClientAreaBox rectangle to get the actual rectangle coordinates.

Events

Unit ImageEnView

TImageEnView

DECLARATION

```
property OnDrawLayerBox: TIEDrawLayerBoxEvent;
```

DESCRIPTION

This event occurs when a layer box must be painted. If you handle this event the default behavior is disabled. ABitmap is the destination bitmap and layer is the layer index that we are drawing. Use Layers[].ClientAreaBox rectangle to get actual rectangle coordinates.

Look at layers demo for more details.

Example

```
procedure TForm1.ImageEnView1DrawLayerBox(Sender:
TObject;
ABitmap: TBitmap; layer: Integer);
begin
  // a green line
  with ABitmap.Canvas do
  begin
    Pen.Style := psSolid;
    Pen.Width := 2;
    Pen.mode := pmCopy;
    Pen.Color := clGreen;
    Brush.Style := bsClear;
  with
    TIELayer(ImageEnView1.Layers[layer]).ClientAreaBox do
      Rectangle(Left-1, Top-1, Right+1,
        Bottom+1);
     end;
end;
```

Unit ImageEnView
TImageEnView

DECLARATION

```
property OnDrawLayerGrip: TIEDrawLayerGrip;
```

DESCRIPTION

This event occurs when a layer grip must be painted. If you handle this event the default behavior is disabled. ABitmap is the destination bitmap and layer is the layer index that we are drawing. Rect specifies the actual grip rectangle.

Look at layers demo for more details.

Example

```
procedure
Tfmain.ImageEnView1DrawLayerGrip(Sender: TObject;
ABitmap: TBitmap; layer, grip: Integer; rect:
TRect);
  begin
    with ABitmap.Canvas do
    begin
      Pen.Style := psSolid;
      Pen.Mode := pmCopy;
      Pen.Color := clGreen;
      Brush.Style := bsSolid;
      Brush.Color := $0000FF00;
      with rect do
        Ellipse(Left-1,Top-1,Right+1,Bottom+1);
    end;
end;
```

Unit ImageEnView
TImageEnView

DECLARATION

```
property OnDrawPolygon: TIEOnDrawPolygon;
```

DESCRIPTION

OnDrawPolygon occurs whenever a polygon is painted.

Unit ImageEnView

TImageEnView

DECLARATION

```
property OnDShowEvent: TNotifyEvent;
```

DESCRIPTION

OnDShowEvent occurs when one o more events are ready.
You should call IO.DShowParams.GetEventCode until it returns false (no more event available).

Example

```
procedure TForm1.ImageEnView1DShowEvent;
var
iEvent: integer;
begin
  while
  ImageEnView1.IO.DShowParams.GetEventCode(IEvent) do
  case IEvent of
    IEEC_COMPLETE:
    begin
      ... end of stream!
    end;
  end;
end;
```

Unit ImageEnView

TImageEnView

DECLARATION

```
property OnDShowNewFrame: TNotifyEvent;
```

DESCRIPTION

OnDShowNewFrame occurs when a new frame is ready to be read. This event is active when you are using IO.DShowParams and EnableSampleGrabber is True.

Example

```
procedure
TForm1.ImageEnView1DShowNewFrame(Sender:
TObject);
begin
  // copy current sample to ImageEnView bitmap
   ImageEnView1.IO.DShowParams.GetSample(ImageEnView1.IEBitmap);
  // refresh ImageEnView1
  ImageEnView1.Update;
end;
```

Unit ImageEnView
TImageEnView

DECLARATION

```
property OnFinishWork: TNotifyEvent;
```

DESCRIPTION

This event occurs whenever an image processing task or input/output terminates. This event is called after the last call to OnProgress, so you can reset here your progress bar.

DECLARATION

```
property OnImageChange: TNotifyEvent;
```

DESCRIPTION

The OnImageChange method notifies of an image change. Method ImageChange calls the event OnImageChange. All image processing effects of TImageEnProc call ImageChange (hence OnImageChange).

Unit ImageEnView

TImageEnView

DECLARATION

```
property OnLayerNotify: TIELayerNotify;
```

DESCRIPTION

The OnLayerNotify event occurs whenever a layer is selected, moved or resized. Only user actions can fire this event.

DECLARATION

```
property OnMouseEnter: TNotifyEvent;
```

DESCRIPTION

OnMouseEnter occurs when the mouse pointer moves over the component. Write OnMouseEnter event handler to take specific action when the user moves the mouse over the component.

DECLARATION

```
property OnMouseInResizingGrip:
TIEMouseInResizingGripEvent;
```

DESCRIPTION

OnMouseInResizingGrip is called whenever the mouse moves and allows you to get the resizing grip where the mouse is positioned.

DECLARATION

```
property OnMouseInSel: TNotifyEvent;
```

DESCRIPTION

OnMouseInSel is called whenever the mouse is inside or on selection contours.

DECLARATION

```
property OnMouseLeave: TNotifyEvent;
```

DESCRIPTION

OnMouseLeave occurs when the mouse pointer moves off of the component. Write OnMouseLeave event handler to take specific action when the user moves the mouse off the component.

DECLARATION

```
property OnPaint: TNotifyEvent;
```

DESCRIPTION

OnPaint occurs when the component is redrawn. Use OnPaint to perform special processing when the component is redrawn.

Unit ImageEnView
TImageEnView

DECLARATION

```
property OnProgress: TIEProgressEvent;
```

DESCRIPTION

The OnProgress event is called when image processing or input/output operations are executed.

DECLARATION

```
property OnSaveUndo: TIESaveUndoEvent;
```

DESCRIPTION

OnSaveUndo is called after SaveUndo is called by user action or by code. Source is the same parameter used calling SaveUndo.

Unit ImageEnView
TImageEnView

DECLARATION

```
property OnSelectionChange: TNotifyEvent;
```

DESCRIPTION

The OnSelectionChange event is called whenever user finishes changing the current selection. Other methods, as AddSelPoint or Select do not call the OnSelectionChange event.

Example

```
// Real time copy of user selected region to ImageEnView1 component
ImageEnView1.Clear;
ImageEnView1.CopyFromPolygon(ImageEnView2.Bitmap,
ImageEnView1.PolySelPoints^, ImageEnView1.PolySelCount, Point(0,0));

// An enhancement of above example: display current user selection as
zoomed in ImageEnView1
ImageEnView1.Clear;
ImageEnView1.CopyFromPolygon(ImageEnView2.Bitmap,
ImageEnView1.PolySelPoints^, ImageEnView1.PolySelCount, Point(0,0));
ImageEnView1.Zoom:=200;
```

Unit ImageEnView

TImageEnView

DECLARATION

```
property OnSelectionChanging: TNotifyEvent;
```

DESCRIPTION

OnSelectionChanging is called whenever a user changes the current selection. Other methods, such as AddSelPoint or Select do not call the OnSelectionChanging event.

DECLARATION

```
property OnSetCursor: TIESetCursorEvent;
```

DESCRIPTION

OnSetCursor occurs whenever mouse cursor needs to be changed. You can set you custom cursor setting Cursor parameter. This event could be called multiple times for the same cursor shape.

DECLARATION

```
property OnSpecialKey: TIESpecialKeyEvent;
```

DESCRIPTION

OnSpecialKey is called whenever a special key is pressed. Special keys are arrows, “home”, “end”, etc. If you have problems to receive OnSpecialKey events, it is suggested to put the TImageEnView (or inherited components) on a TPanel instead of TForm.

DECLARATION

```
property OnTransitionPaint: TIEOnTransitionPaint;
```

DESCRIPTION

This event occurs just before a new transition frame is painted. Step assumes values from 0 to 1024.

DECLARATION

```
property OnTransitionStep:
TIETransitionStepEvent;
```

DESCRIPTION

This event occurs just before a new transition frame is painted. Step assumes values from 0 to 1024.

DECLARATION

```
property OnTransitionStop: TNotifyEvent;
```

DESCRIPTION

This event occurs when the transition (runs using RunTransition) has finished its job, that is the transition is finished.

Unit ImageEnView

TImageEnView

DECLARATION

```
property OnViewChange: TViewChangeEvent;
```

DESCRIPTION

OnViewChange notifies of a Zoom, ViewX or ViewY properties change.

DECLARATION

```
property OnViewChanging: TViewChangeEvent;
```

DESCRIPTION

Notifies that Zoom, ViewX or ViewY is going to be changed (but could not change).

DECLARATION

```
property OnVirtualKey: TIEVirtualKeyEvent;
```

DESCRIPTION

This event occurs whenever the component receives WM_KEYDOWN, WM_SYSKEYDOWN, WM_KEYUP or WM_SYSKEYUP message.

Unit ImageEnView
TImageEnView

DECLARATION

```
property OnZoomIn: TIEZoomEvent;
```

DESCRIPTION

An OnZoomIn event occurs whenever the user zooms-in on the current image. Applications can customize zoom value changing the NewZoom parameter.

Example

```
// lock zoom-in to 50
procedure TForm1.ImageEnView1ZoomIn(Sender:
Tobject; var NewZoom: Double);
begin
  NewZoom := 50;
end;
```

Unit ImageEnView
TImageEnView

DECLARATION

```
property OnZoomOut: TIEZoomEvent;
```

DESCRIPTION

An OnZoomOut event occurs whenever the user zooms-out on the current image. Applications can customize the zoom value changing the NewZoom parameter.

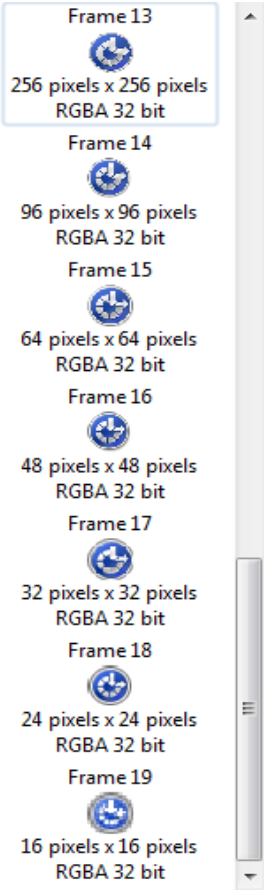
Example

```
// lock zoom-out to 200
procedure Tform1.ImageEnView1ZoomOut(Sender:
Tobject; var NewZoom: Double);
begin
  NewZoom := 200;
end;
```

Chapter 4

TImageEnMView

Chapter 4. ImageEnMView



The TImageEnMView component is the similar to TImageEnView, but it can handle multiple images. TImageEnMView can display images in rows, in columns, in a grid or in a single frame. The images can be animated. If you choose to show a single frame, you can view an animated sequence (as a Gif or an AVI). Each image has assigned a delay time.

The images can be stored as thumbnails (a sub-resampled image of the original), or stored as the full image. Images may be loaded when displayed (you have to specify only the file name), or upon request (whenever an image is to be shown, an event is generated).

Unit ImageEnMView
TImageEnMView

If you choose, the user can select an image. Selected images will have a bordered frame. You can also specify a custom function whenever an image is shown (i.e., to paint the image index near the image).

Note: Users often attach a TImageEnIO component to TImageEnMView component. This is not correct. The TImageEnMIO component must be attached only to a TImageEnMView component, or you can use embedded MIO object.

Display images "on demand" with TImageEnMView

1) If you have a directory where are all files just write:

```
ImageEnMView1.FillFromDirectory('c:\myimages');
```

2) When you add a new image just set ImageFileName[] index, and ImageEn will load automatically specified file when needed.

Example

```
idx:=ImageEnMView1.AppendImage;
ImageEnMView1.ImageFileName[idx]:='first.jpg';
```

3) When you add a new image just set the ImageID[] property. You have to create by hand an array of filenames where to get images.

Example

```
var
files: array [0..1] of string;
begin
files[0] := 'first.jpg';
files[1] := 'second.jpg';
ImageEnMView1.ImageID[ ImageEnMView1.AppendImage
] := 0;
ImageEnMView1.ImageID[ ImageEnMView1.AppendImage
] := 1;
end;
```

You have also to create OnImageIDRequest event, on this you can write:

```
procedure TForm1.OnImageIDRequest(Sender:
TObject; ID: integer; var Bitmap: TBitmap);
var
iImageEnIO: TImageEnIO;
begin
iImageEnIO:= TImageEnIO.Create(self);
iImageEnIO.AttachedBitmap := bmp;
// bmp is a TBitmap object, defined at class
// level (must exists after the OnImageIDRequest
// exits)
iImageEnIO.LoadFromFile(files[ID]);
iImageEnIO.free;
Bitmap := bmp;
end;
```

4) If the images are frames of a media file (like AVI, MPEG, etc..) you can write:

```
ImageEnMView1.LoadFromFileOnDemand('film.mpeg');
```

To print all images in TImageEnMView connect a TImageEnIO to a TImageEnMView. TImageEnIO.PrintImage will print the selected image (use SelectedImage to change current selected image). In this way you can print all pages:

```
for i:= 0 to ImageEnMView1.ImageCount-1 do
begin
  ImageEnMView1.SeletedImage := i;
  ImageEnIO.PrintImage;
end;
```

You can also use the predefined dialog of TImageEnMView component. Just write:

```
ImageEnMView1.MIO.DoPrintPreviewDialog;
```

Unit ImageEnMView
TImageEnMView

Copy A TImageEnMView Image To TImageEnView

When a user selects a page in TImageEnMView you can display this image in a TImageEnView by handling the OnImageSelect event. To transfer current selected TImageEnMView image to TImageEnView just use the Assign method:

```
procedure TForm1.ImageEnMView1ImageSelect(Sender:
TObject; idx: Integer);
begin
  ImageEnView1.Assign(ImageEnMView1.Bitmap);
end;
```

Unit ImageEnMView

TImageEnMView

Methods and Properties

Display

Animation	BackgroundStyle	CenterFrame
ClearImageCache	DisplayImageAt	DisplayMode
EnableAlphaChannel	FlatScrollBars	GradientEndColor
GridWidth	HighlightColor	HighlightTextColor
HorizBorder	LockPaint	LockPaintCount
LockUpdate	LockUpdateCount	MaximumViewX
MaximumViewY	SetChessboardStyle	SetPresetThumbnailFrame
SetViewXY	SoftShadow	Style
UnLockPaint	UnLockUpdate	VertBorder
ViewX	ViewY	VisibleFrame
WallPaper	WallPaperStyle	

Image Editing

AppendImage	AppendImage2	AppendSplit
Clear	CreateMorphingSequence	DeleteImage
InsertImageEx	InsertImage	MoveImage
RemoveBlankPages	Sort	

Unit ImageEnMView
TImageEnMView

Image Access and Copying

Bitmap	CopyToIEBitmap	GetBitmap
GetTIEBitmap	IEBitmap	PrepareSpaceFor
ReleaseBitmap	SetIEBitmap	SetImageEx
SetImageRect	SetImage	UpdateImage

Image Info

ImageBitCount	ImageCol	ImageCount
ImageFileName	ImageHeight	ImageID
ImageOriginalHeight	ImageOriginalWidth	ImageRow
ImageTag	ImageUserPointer	ImageWidth
ImageX	ImageY	

Image Text

DefaultBottomTextFont	DefaultInfoTextFont	DefaultTopTextFont
ImageBottomText	ImageInfoText	ImageTopText
ShowText		

Unit ImageEnMView
TImageEnMView

Thumbnails Appearance

BottomGap

DrawImageBackground

FillThumbnail

ImageBackground

ThumbHeight

ThumbnailDisplayFilter

ThumbnailFrameRect

ThumbnailFrameSelected

ThumbnailFrame

ThumbnailResampleFilter

ThumbnailsBackground

ThumbnailsBackgroundStyle

ThumbnailsBorderColor

ThumbnailsBorderWidth

ThumbnailsInternalBorderColor

ThumbnailsInternalBorder

ThumbsRounded

ThumbWidth

UpperGap

Unit ImageEnMView
TImageEnMView

Input/Output

EnableAdjustOrientation EnableImageCaching EnableLoadEXIFThumbnails

EnableResamplingOnMinor FillFromDirectory GetImageToFile

GetImageToStream ImageCacheSize ImageCacheUseDisk

JobsRunning JobsWaiting LoadFromFileOnDemand

LoadIconOnUnknownForma LoadSnapshot LookAhead

MaintainInvisibleImages MIO ReloadImage

RemoveCorrupted SaveSnapshot SetImageFromFile

SetImageFromStream ThreadPoolSize

Selections

BeginSelectImages CenterSelected CheckThumbBoundsOnSele

DeleteSelectedImages DeSelect EnableMultiSelect

EndSelectImages IsSelected MoveSelectedImagesTo

MultiSelectedImages MultiSelectedimagesCount MultiSelecting

MultiSelectionOptions MultiSelectSortList SelectAll

SelectedImage SelectionAntialiased SelectionColor

SelectionStyle SelectionWidth SelectionWidthNoFocus

SelectSeek TrackMouseSelection UnSelectImage

VisibleSelection

Unit ImageEnMView
TImageEnMView

User Interaction

GetImageVisibility	HScrollBarParams	ImageAtGridPos
ImageAtPos	InsertingPoint	IsVisible
KeyInteract	MouseInteract	ScrollBars
ScrollBarsAlwaysVisible	VScrollBarParams	

Animations and Transitions

ImageDelayTime	Playing	PlayLoop
TransitionDuration	TransitionEffect	TransitionRunning

Others

CalcGridWidth	GetLastOp	GetLastOpIdx
IEBeginDrag	IEEndDrag	ImageEnVersion
Proc	StoreType	Update
UpdateCoords		

Unit ImageEnMView
TImageEnMView

Events

OnAllDisplayed	OnBeforeImageDrawEx	OnBeforeImageDraw
OnDrawProgress	OnFinishWork	OnImageAtPos
OnImageDeselect	OnImageDraw	OnImageDraw2
OnImageIDRequestEx	OnImageIDRequest	OnImageSelect
OnIOProgress	OnMouseEnter	OnMouseLeave
OnPlayFrame	OnViewChange	OnWrongImage

Unit ImageEnMView

TImageEnMView

Display

Declaration

```
property Animation: TIEAnimation;
```

Description

Animation specifies the animation to use. Default is nil (that is “no animation”).

Example

```
ImageEnMView1.Animation :=
TIEHorizontalFlow.Create();
```

Declaration

```
property BackgroundStyle: TIEBackgroundStyle;
```

Description

BackgroundStyle specifies the background style. The background is the component region not filled by the image or thumbnails.

Example

```
ImageEnMView1.BackgroundStyle:=iebsChessboard;
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure CenterFrame;
```

DESCRIPTION

CenterFrame shows the current frame in the center.
This method uses VisibleFrame to get/set current frame.

DECLARATION

```
procedure ClearImageCache(idx: integer);
```

DESCRIPTION

The ClearImageCache method clears the cache for image idx. Image caching allows you to speed up the thumbnails painting, saving each drawn image to a cache. You should call this method only if you have refreshing problems. See also EnableImageCaching.

Unit ImageEnMView

TImageEnMView

DECLARATION

```
procedure DisplayImageAt(imageIndex: integer; x,
y: integer);
```

DESCRIPTION

DisplayImageAt scrolls to make image visible at specified position.

Parameter	Description
imageIndex	Index of the image to display.
x	Client area horizontal offset.
y	Client area vertical offset.

Example

```
// make image 10 visible at 0,0
ImageEnMView1.DisplayImageAt(10, 0, 0);
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property DisplayMode: TIEMDisplayMode;
```

DESCRIPTION

DisplayMode can be mdGrid or mdSingle.
To show images in a single row set DisplayMode to mdGrid, and GridWidth to 0. To show images in a single column set DisplayMode to mdGrid, and GridWidth to 1.

DECLARATION

```
property EnableAlphaChannel: Boolean;
```

DESCRIPTION

Set EnableAlphaChannel to True to enable thumbnails' alpha channel (it is also requested to show soft shadows).

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property FlatScrollBars: Boolean;
```

DESCRIPTION

FlatScrollBars specifies whether the component's scroll bars are flat. ImageEn only supports flat scroll bars if the system has version 472 or later of comctl32.dll and Delphi 5 or later and C++Builder 5 or later.

DECLARATION

```
property GradientEndColor: TColor
```

DESCRIPTION

GradientEndColor specifies the ending color of the gradient when BackgroundStyle is iebsGradient.

DECLARATION

```
property GridWidth: integer;
```

DESCRIPTION

GridWidth is the number of images per row. This property is active only when DisplayMode is mdGrid.

Valid GridWidth values are:

Value	Description
-1	GridWidth adapted to the component width
0	Only one row
>0	GridWidth images per row

DECLARATION

```
property HighlightColor: TColor;
```

DESCRIPTION

When SelectionStyle = iessACD, this property specifies the color used for backgrounds. Default is clHighlight.

DECLARATION

```
property HighlightTextColor: TColor;
```

DESCRIPTION

When SelectionStyle=iessACD, this property specifies the color used for texts. Default is clHighlightText.

ImageEnMView

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property HorizBorder: integer;
```

DESCRIPTION

HorizBorder is the horizontal distance between two images.
See also VertBorder.

DECLARATION

```
procedure LockPaint;
```

DESCRIPTION

The LockPaint method increases the lock counter's value.
Use UnLockPaint to unlock.

DECLARATION

```
property LockPaintCount: integer;
```

DESCRIPTION

Returns lock painting state. 0=no lock, >0 locking.
LockPaint increases LockPaintCount, UnLockPaint decreases it.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure LockUpdate;
```

DESCRIPTION

The LockUpdate method increases the lock counter's value. Use UnLockUpdate to unlock.

DECLARATION

```
property LockUpdateCount: integer;
```

DESCRIPTION

Returns lock update state. 0=no lock, >0 locking. LockUpdate increases LockUpdateCount, UnLockUpdate decreases it.

DECLARATION

```
property MaximumViewX: integer;
```

DESCRIPTION

MaximumViewX returns the maximum value that you can assign to the ViewX property.

See also
MaximumViewY.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property MaximumViewY: integer;
```

DESCRIPTION

MaximumViewY returns the maximum value that you can assign to the ViewY property.

See also
MaximumViewX.

DECLARATION

```
procedure SetChessboardStyle(Size: integer;
BrushStyle: TBrushStyle);
```

DESCRIPTION

SetChessboardStyle specifies size and brush of the chessboard background (see BackgroundStyle).

Parameter	Description
Size	Specifies the box size (default 16)
BrushStyle	Specifies the brush style of the boxes (default bsSolid)

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure SetPresetThumbnailFrame(PresetIndex:
integer; UnSelectedColor: TColor; SelectedColor:
TColor);
```

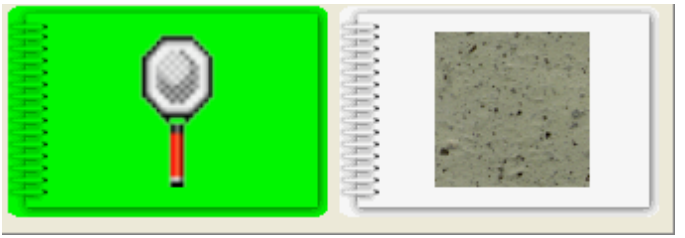
DESCRIPTION

SetPresetThumbnailFrame specifies a background to draw for each thumbnail. PresetIndex specifies a preset image index. Currently values from 0 to 3 are available. UnSelectedColor specifies how modify the preset image when it is unselected. SelectedColor specifies how modify the preset image when it is selected.

Examples

```
ImageEnMView1.SetPresetThumbnailFrame(0, clWhite,
clGreen);
```

```
ImageEnMView1.SetPresetThumbnailFrame(2, clWhite,
clGreen);
```



See demo

viewers\thumbnails2

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure SetViewXY(x, y: integer);
```

DESCRIPTION

SetViewXY sets ViewX and ViewY in one step.

DECLARATION

```
property SoftShadow: TIEVSoftShadow;
```

DESCRIPTION

SoftShadow allows painting a shadow under the thumbnails.

Example

```
ImageEnMView1.EnableAlphaChannel := True;
ImageEnMView1.SoftShadow.Enabled := True;
```

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
TIEVSoftShadow
```

DESCRIPTION

The TIEVSoftShadow class specifies how a shadow will be rendered.

Fields

DECLARATION

```
Enabled: Boolean;
```

DESCRIPTION

Enabled enables/disables the shadow.

DECLARATION

```
Radius: double;
```

DESCRIPTION

Radius sets the radius of the shadow.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
OffsetX: integer;
```

DESCRIPTION

OffsetX specifies the horizontal relative position of the shadow.

DECLARATION

```
OffsetY: integer;
```

DESCRIPTION

OffsetY specifies the vertical relative position of the shadow.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property Style: TIEMStyle;
```

DESCRIPTION

Style specifies the thumbnails style. It can be flat or 3D.

Example

```
ImageEnMView1.Style := iemsFlat;
```


```
ImageEnMView1.Style := iemsACD;
```

Unit ImageEnMView

TImageEnMView

DECLARATION

```
TIEMStyle = (iemsFlat, iemsACD);
```

DESCRIPTION

Value	Description
iemsFlat	flat style (old ImageEn style)
iemsACD	3D style (default)

DECLARATION

```
function UnLockPaint: integer;
```

DESCRIPTION

Use the UnLockPaint method to decrease the lock counter's value locked using LockPaint. If the lock count is zero, then the Update method is called. UnLockPaint returns the lock count.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
function UnLockUpdate: integer;
```

DESCRIPTION

Use the UnLockUpdate method to decrease the lock counter's value locked using LockUpdate. If the lock count is zero, then the Update method is called.

UnLockUpdate returns the lock count.

DECLARATION

```
property VertBorder: integer;
```

DESCRIPTION

VertBorder is the vertical distance between two images.
See also HorizBorder.

DECLARATION

```
property ViewX: integer;
```

DESCRIPTION

ViewX is the first column displayed on left-upper side of the component. You can set ViewX to simulate horizontal scroll-bar movement.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ViewY: integer;
```

DESCRIPTION

ViewY is the first row displayed on left-upper side of the component. You can set ViewY to simulate vertical scroll-bar movement.

See also
ViewX.

DECLARATION

```
property VisibleFrame: integer;
```

DESCRIPTION

VisibleFrame represents the visible image when DisplayMode is dmSingle or Playing is true.

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property WallPaper: TPicture;
```

DESCRIPTION

The WallPaper property sets a background image under the thumbnails. Use WallPaperStyle to specify how to paint the wallpaper.

DECLARATION

```
property WallPaperStyle: TIEWallPaperStyle;
```

DESCRIPTION

WallPaperStyle specifies how to paint the wallpaper. WallPaper specifies the image to use.

Unit ImageEnMView
TImageEnMView

DECLARATION

TIEWallPaperStyle = (iewoNormal, iewoStretch, iewoTile);

DESCRIPTION

Value	Description
iewoNormal	paint the wall paper image at the top left side
iewoStretch	paint the wall paper stretching to the component size
iewoTile	paint the wall paper tiled to the component size

ImageEnMView

Unit ImageEnMView

TImageEnMView

Image editing

Declaration

```
function AppendImage: integer;
function AppendImage(const FileName:string):
integer;
function AppendImage(Stream: TStream): integer;
function AppendImage(Bitmap: TIEBitmap): integer;
```

Description

AppendImage appends a new image at last position in the list and returns the new image position. First overload of AppendImage doesn't create the bitmap. Others load image from a file or a stream.

Example

```
ImageEnView1.IO.LoadFromFile('000.tif');
idx:=ImageEnMView1.AppendImage;
ImageEnMView1.SetImage(idx,ImageEnView1.Bitmap);

// the same job...
ImageEnMView1.AppendImage('000.tif');
```

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
function AppendImage2(Width, Height: integer;
PixelFormat: TIEPixelFormat): integer;
```

DESCRIPTION

AppendImage2 performs the same operations of AppendImage and also creates a new image with specified size and pixelformat.

Example

```
ImageEnMView1.AppendImage2(256,256,ie24RGB);
```

Unit ImageEnMView

TImageEnMView

DECLARATION

```
function AppendSplit(SourceGrid: TIEBitmap;
cellWidth: integer; cellHeight: integer;
maxCount: integer = 0):integer;
```

DESCRIPTION

Splits source image in cell of specified size and add each cell. Returns added images count.

Parameter	Description
SourceGrid	Source bitmap containing cells to split.
cellWidth	Width of a cell.
cellHeight	Height of a cell.
maxCount	Maximum number of cells to add. 0 = all suitable cells.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
function CalcGridHeight(): integer;
```

DESCRIPTION

CalcGridHeight calculates the grid height (the number of rows of the grid).

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure Clear;
```

DESCRIPTION

Clear removes all images.

DECLARATION

```
procedure CreateMorphingSequence(Source:
TImageEnVect; Target: TImageEnVect; FramesCount:
integer);
```

DESCRIPTION

This method creates a sequence of frames which are the transformation of the Source image to Target image. Images PixelFormat must be ie24RGB (true color) and have the same size. Also Source and Target must contain iekLINE objects (the same number) which describe the transformation. You should create line objects in the same order on both TImageEnVect components. FramesCount specifies the number of frames to create.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure DeleteImage(idx: integer);
```

DESCRIPTION

DeleteImage deletes idx image and frees bitmap memory.

DECLARATION

```
procedure InsertImageEx(idx: integer);
```

DESCRIPTION

InsertImageEx inserts a new image in idx position (0 is the first). InsertImageEx doesn't create the bitmap and doesn't make the image selected (the only difference from InsertImage method).

Example

```
ImageEnView1.IO.LoadFromFile('000.tif');
ImageEnMView1.InsertImageEx(0);
ImageEnMView1.SetImage(0,ImageEnView1.Bitmap);
```

See also
InsertImage method
InsertImageEx method
DeleteImage method
AppendImage method

Unit ImageEnMView

TImageEnMView

DECLARATION

```
procedure InsertImage(idx: integer);
procedure InsertImage(Idx : integer; const
FileName : string);
procedure InsertImage(Idx : integer; Stream :
TStream);
procedure InsertImage(Idx : integer; Bitmap :
TIEBitmap);
procedure InsertImage(Idx : integer; Bitmap :
TBitmap);
procedure InsertImage(Idx : integer; Width,
Height : integer; PixelFormat : TIEPixelFormat =
ie24RGB);
```

DESCRIPTION

Inserts or appends a new image in idx position (0 is the first). InsertImage doesn't create the bitmap.

Example

```
ImageEnView1.IO.LoadFromFile('000.tif');
ImageEnMView1.InsertImage(0);
ImageEnMView1.SetImage(0, ImageEnView1.Bitmap);
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure MoveImage(idx: integer; destination:
integer);
```

DESCRIPTION

MoveImage moves an image from the index position idx to destination position. If the destination index is equal to or greater than the image count, the idx image is moved to the position immediately after the last image.

Example

```
// exchange first and second images
ImageEnMView1.MoveImage(0,1);

// move first image after last image
ImageEnMView1.MoveImage(0,ImageEnMView1.ImageCoun
t);
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
function RemoveBlankPages(Tolerance: double;
Complete: Boolean; LeftToRight: Boolean):integer;
```

DESCRIPTION

Detects images with a single color (ie a blank page) and remove them.

Parameter	Description
Tolerance	Controls the amount if purity in order to detect a blank page. Values can range from 0.0 up to 1.0. For example when tolerance is 0.1 then 10% of pixels can have different colors.
Complete	If true all images are checked. Otherwise the check stops when the first non-blank image has found.
LeftToRight	If true the scan proceeds from left to right (otherwise it proceeds from right to left).

Example

```
// remove last blank pages
ImageEnMView1.RemoveBlankPages(0.0, false,
false);
```

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure Sort(Compare:
TIEImageEnMViewSortCompare);
procedure Sort(Compare:
TIEImageEnMViewSortCompareEx);
```

DESCRIPTION

Sort sorts all images using specified comparison function.

Example

```
function xcompare(i1,i2: integer): integer;
var
iString1,1String2: integer;
begin
  with Form1.ImageEnMView1 do
  begin
    iString1 :=
      ImageOriginalWidth[i1]*ImageOriginalHeight[i1];
    iString2 :=
      ImageOriginalWidth[i2]*ImageOriginalHeight[i2];
  end;
  if iString1 < iString2 then
    result := -1
  else if iString1> iString2 then
    result := 1
  else
    result := 0;
end;
```

See next page for sort…

Unit ImageEnMView

TImageEnMView

```
// Sort By Size
procedure TForm1.Button1Click(Sender: TObject);
begin
  ImageEnMView1.Sort(xcompare);
end;
```

ImageEnMView

Unit ImageEnMView

TImageEnMView

Image access and copying

DECLARATION

```
property Bitmap: TBitmap;
```

DESCRIPTION

Bitmap is the currently selected image as TBitmap object.

DECLARATION

```
procedure CopyToIEBitmap(idx: integer; bmp:
TIEBitmap);
```

DESCRIPTION

CopyToIEBitmap method copies the specified idx image to the destination IEBitmap object.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
function GetBitmap(idx: integer): TBitmap;
```

DESCRIPTION

GetBitmap creates a TBitmap object from the image contained in idx index. Each change you make to the bitmap will be visible after you have called the Update method. You have to call ReleaseBitmap to free the TBitmap object.

See also
GetTIEBitmap.

Example

```
bmp := ImageEnMView1.GetBitmap(0);
bmp.SaveToFile('alfa.bmp');
ImageEnMView1.ReleaseBitmap(0);
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
function GetTIEBitmap(idx: integer): TIEBitmap;
```

DESCRIPTION

GetTIEBitmap creates a TIEBitmap object from the image contained at idx index. Each change you make in the bitmap will be visible after you have called Update method.
You have to call ReleaseBitmap to free the TIEBitmap object.

Example

```
bmp := ImageEnMView1.GetTIEBitmap(0);
// bmp must be TIEBitmap type
. . .
ImageEnMView1.ReleaseBitmap(0);
```

DECLARATION

```
property IEBitmap: TIEBitmap;
```

DESCRIPTION

IEBitmap is the currently selected image as a TIEBitmap object.

Unit ImageEnMView

TImageEnMView

DECLARATION

```
procedure PrepareSpaceFor(Width, Height: integer;
Bitcount: integer; ImageCount: integer);
```

DESCRIPTION

PrepareSpaceFor allocates enough space on a temporary file for ImageCount images of size Width*Height*BitCount. Call this method to improve performance only when you plan to add many images of the same size.

DECLARATION

```
procedure ReleaseBitmap(idx: integer;
saveChanges: Boolean = true);
```

DESCRIPTION

ReleaseBitmap releases the bitmap created with GetBitmap or GetTIEBitmap method.

Parameter	Description
idx	The image index to release.
saveChanges	If true (default) the bitmap will be written in the cache.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure SetIEBitmap(idx: integer; srcImage:
TIEBaseBitmap);
```

DESCRIPTION

SetIEBitmap sets the image assigned to idx index. The srcImage bitmap is copied internally; therefore you can free srcImage after SetIEBitmap.

Example

```
ImageEnView1.IO.LoadFromFile('000.tif');
ImageEnMView1.InsertImageEx(0);
ImageEnMView1.SetIEBitmap(0,ImageEnView1.IEBitmap);
```

Unit ImageEnMView

TImageEnMView

DECLARATION

```
procedure SetImageEx(idx: integer; srcImage:
TBitmap);
```

DESCRIPTION

SetImageEx sets the image assigned to idx index. The srcImage bitmap is copied internally; therefore you can free srcImage after SetImageEx. SetImageEx doesn't call Update (the only difference from SetImage).

Example

```
ImageEnView1.IO.LoadFromFile('000.tif');
ImageEnMView1.InsertImageEx(0);
ImageEnMView1.SetImageEx(0,ImageEnView1.Bitmap);
ImageEnMView1.Update; // no needed on SetImage()
```

DECLARATION

```
procedure SetImageRect(idx: integer; srcImage:
TBitmap; x1, y1, x2, y2: integer);
procedure SetImageRect(idx: integer; srcImage:
TIEBitmap; x1, y1, x2, y2:integer);
```

DESCRIPTION

SetImageRect sets the image assigned to idx index. The rectangle x1, y1, x2, y2 of srcImage bitmap is copied internally. After SetImageRect you can free srcImage bitmap.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure SetImage(idx: integer; srcImage:
TBitmap);
procedure SetImage(idx: integer; width, height:
integer; PixelFormat: TIEPixelFormat);
```

DESCRIPTION

SetImage sets the image assigned to idx index. The srcImage bitmap is copied internally; therefore you can free srcImage after SetImage.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure UpdateImage(idx: integer);
```

DESCRIPTION

Call UpdateImage to redraw only the idx image.

You have to update TImageEnMView component (with Update or UpdateImage methods) whenever an image (Bitmap) is changed.

Example

```
// this draws a rectangle on image 3
var
  bmp: TBitmap;
begin
  bmp := ImageEnMView1.GetBitmap(3);
  bmp.canvas.rectangle(0,0,10,10);
  ImageEnMView1.ReleaseBitmap(3);
  UpdateImage(3);
end;
```

Unit ImageEnMView

TImageEnMView

Image info

DECLARATION

```
property ImageBitCount[idx: integer]:integer;
```

DESCRIPTION

ImageBitCount returns the bit count of the specified idx image. ImageBitCount can be:
1 : black/white image
24 : true color image

DECLARATION

```
property ImageCol[idx: integer]: integer;
```

DESCRIPTION

ImageCol returns the column where is the image idx. ImageCol[] is 0 for the first-left image, 1 for the second-left image, etc.

Read only

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ImageCount: integer;
```

DESCRIPTION

ImageCount is the number of images stored in the TImageEnMView component.

Read-only

DECLARATION

```
property ImageFileName[idx: integer]: WideString;
```

DESCRIPTION

Specifies the file from where TImageEnMView will obtain the image when it is to be shown. This property contains the full file path or a special form to load multi-page files. The special form is 'fullfilepath::imageindex'.

Example

```
// load image1.jpg as thumbnail 0 and image2.jpg as
thumbnail 1
ImageEnView1.ImageFileName[0]:='c:\image1.jpg';
ImageEnView1.ImageFileName[1]:='c:\image2.jpg';
// load frame 0 and 1 from input.mpeg
ImageEnView1.ImageFileName[0]:='c:\input.mpeg::0';
ImageEnView1.ImageFileName[1]:='c:\input.mpeg::1';
```

ImageEnMView

Example

This example inserts two images. The first is loaded from 'one.tif' and the second from 'two.tif'. The images are loaded only when these are shown (otherwise these are freed).

```
ImageEnMView1.InsertImage(0);
ImageEnMView1.ImageFileName[0]:='one.tif';
ImageEnMView1.InsertImage(1);
ImageEnMView1.ImageFileName[1]:='two.tif';
```

DECLARATION

```
property ImageHeight[idx: integer];
```

DESCRIPTION

ImageWidth and ImageHeight are the sizes of the image idx.

DECLARATION

```
property ImageID[idx: integer]: integer;
```

DESCRIPTION

ImageID associates a numeric ID with idx image. This ID is returned to OnImageIDRequest event.

Unit ImageEnMView
TImageEnMView

Declaration

```
property ImageOriginalHeight[idx: integer]:
integer;
```

Description

ImageOriginalHeight get/set the original height of the specified idx image. This is useful only when you store images as thumbnails and need to know the original image size.

See also
ImageOriginalWidth.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ImageOriginalWidth[idx: integer]: integer;
```

DESCRIPTION

ImageOriginalWidth get/set the original width of the specified idx image. This is useful only when you store images as thumbnails and need to know the original image size.

See also ImageOriginalHeight.

DECLARATION

```
property ImageRow[idx: integer]: integer;
```

DESCRIPTION

ImageRow returns the row where is the image idx. ImageRow[] is 0 for the first-upper image, 1 for the second-upper image, etc.

Read only

ImageEnMView

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property ImageTag[idx: integer]: integer;
```

DESCRIPTION

ImageTag associates a numeric integer value with idx image. This is a user value and is not used by TImageEnMView. This value is not saved or loaded from file/streams and copy/pasted to clipboard.

DECLARATION

```
property ImageUserPointer[idx: integer]:pointer;
```

DESCRIPTION

ImageTag associates a pointer with idx image. This is a user value and is not used by TImageEnMView. ImageUserPointer value is not saved or loaded from file/stream and copy/pasted to clipboard.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ImageWidth[idx: integer];
```

DESCRIPTION

ImageWidth and ImageHeight are the sizes of the image idx.

DECLARATION

```
property ImageX[idx: integer]: integer;
```

DESCRIPTION

ImageX returns the column (in pixels) where the image will be shown.

Read-only

ImageEnMView

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property ImageY[idx: integer]: integer;
```

DESCRIPTION

ImageY returns the row (in pixels) where the image will be shown.

Read-only

Image text

DECLARATION

```
property DefaultBottomTextFont: TFont;
```

DESCRIPTION

Specifies the default font for bottom text (see also ImageBottomText).

See also:
DefaultInfoTextFont
DefaultTopTextFont

Example

```
ImageEnMView1.DefaultBottomTextFont.Height := 14;
ImageEnMView1.DefaultBottomTextFont.Name :=
'Arial';
ImageEnMView1.FillFromDirectory('pictures');
```

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property DefaultInfoTextFont: TFont;
```

DESCRIPTION

Specifies the default font for info text (see also ImageInfoText).

See also:
DefaultBottomTextFont
DefaultTopTextFont

DECLARATION

```
property DefaultTopTextFont: TFont;
```

DESCRIPTION

Specifies the default font for top text (see also ImageTopText).

See also:
DefaultBottomTextFont
DefaultInfoTextFont

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ImageBottomText[idx: integer]: TIEMText;
```

DESCRIPTION

Use ImageTopText, ImageBottomText and ImageInfoText to specify an optional text to write inside the thumbnail idx. TIEMText is an object that specifies how the text will be written. The BottomGap and UpperGap properties will be adjusted to fit the text.

Example

```
ImageEnMView1.ImageTopText[idx].Caption:='Top text';
ImageEnMView1.ImageInfoText[idx].Caption:='Info text';
ImageEnMView1.ImageBottomText[idx].Caption:='Bottom text';
```


```
ImageEnMView1.ImageTopText[idx].Caption:='Top text';
ImageEnMView1.ImageInfoText[idx].Caption:='Info text';
ImageEnMView1.ImageBottomText[idx].Caption:='Bottom text';
ImageEnMView1.ImageBottomText[idx].Background:=clYellow;
ImageEnMView1.ImageBottomText[idx].Font.Color:=clBlue;
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ImageInfoText[idx: integer]: TIEMText;
```

DESCRIPTION

Use ImageTopText, ImageBottomText and ImageInfoText to specify an optional text to write inside the thumbnail idx. TIEMText is an object that specifies how the text will be written. The BottomGap and UpperGap properties will be adjusted to fit the text.

Example

```
ImageEnMView1.ImageTopText[idx].Caption:='Top text';
ImageEnMView1.ImageInfoText[idx].Caption:='Info text';
ImageEnMView1.ImageBottomText[idx].Caption:='Bottom text';
```


```
ImageEnMView1.ImageTopText[idx].Caption:='Top text';
ImageEnMView1.ImageInfoText[idx].Caption:='Info text';
ImageEnMView1.ImageBottomText[idx].Caption:='Bottom text';
ImageEnMView1.ImageBottomText[idx].Background:=clYellow;
ImageEnMView1.ImageBottomText[idx].Font.Color:=clBlue;
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ImageTopText[idx: integer]: TIEMText;
```

DESCRIPTION

Use ImageTopText, ImageBottomText and ImageInfoText to specify an optional text to write inside the thumbnail idx. TIEMText is an object that specifies how the text will be written. The BottomGap and UpperGap properties will be adjusted to fit the text.

Example

```
ImageEnMView1.ImageTopText[idx].Caption:='Top text';
ImageEnMView1.ImageInfoText[idx].Caption:='Info text';
ImageEnMView1.ImageBottomText[idx].Caption:='Bottom text';
```


```
ImageEnMView1.ImageTopText[idx].Caption:='Top text';
ImageEnMView1.ImageInfoText[idx].Caption:='Info text';
ImageEnMView1.ImageBottomText[idx].Caption:='Bottom text';
ImageEnMView1.ImageBottomText[idx].Background:=clYellow;
ImageEnMView1.ImageBottomText[idx].Font.Color:=clBlue;
```

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property ShowText: Boolean;
```

DESCRIPTION

If True (default) text top, bottom and info text is displayed.

Thumbnails appearance

DECLARATION

```
property BottomGap: integer;
```

DESCRIPTION

BottomGap is the distance from image and its border.
You can set BottomGap to reserve space for your painting (with OnImageDraw).

DECLARATION

```
property DrawImageBackground: Boolean;
```

DESCRIPTION

If DrawImageBackground is true, the image background is painted. Otherwise, the component background is painted.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property FillThumbnail: Boolean;
```

DESCRIPTION

When FillThumbnail is true (default), the thumbnail is filled with the background color (ThumbnailsBackgroundStyle).

DECLARATION

```
property ImageBackground: TColor;
```

DESCRIPTION

ImageBackground is the color of the space between images.

DECLARATION

```
property ThumbHeight: integer;
```

DESCRIPTION

ThumbHeight is the height of the thumbnail. If UpperGap and BottomGap are 0, ThumbHeight corresponds to the image height. See also ThumbWidth.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ThumbnailDisplayFilter: TResampleFilter;
```

DESCRIPTION

ThumbnailDisplayFilter specifies a filter to apply when an image (thumbnail) need to be resized. For black/white images it is automatically rfFastLinear.

DECLARATION

```
property ThumbnailFrameRect: TRect;
```

DESCRIPTION

Using ThumbnailFrameSelected and ThumbnailFrame, this property specifies where the image (the thumbnail) will be drawn.

Examples

```
ImageEnMView1.ThumbnailFrame :=
ImageEnViewUnSelected.IEBitmap;
ImageEnMView1.ThumbnailFrameSelected :=
ImageEnViewSelected.IEBitmap;
ImageEnMView1.ThumbnailFrameRect:=Rect(10,10,50,50);
```

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ThumbnailFrameSelected: TIEBitmap;
```

DESCRIPTION

ThumbnailFrameSelected specifies a bitmap to display under the thumbnail when it is selected.

Examples

```
ImageEnMView1.ThumbnailFrame :=
ImageEnViewUnSelected.IEBitmap;
ImageEnMView1.ThumbnailFrameSelected :=
ImageEnViewSelected.IEBitmap;
ImageEnMView1.ThumbnailFrameRect:=Rect(10,10,50,5
0);
```

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ThumbnailFrame: TIEBitmap;
```

DESCRIPTION

ThumbnailFrame specifies a bitmap to display under the thumbnail.

Examples

```
ImageEnMView1.ThumbnailFrame :=
ImageEnViewUnSelected.IEBitmap;
ImageEnMView1.ThumbnailFrameSelected :=
ImageEnViewSelected.IEBitmap;
ImageEnMView1.ThumbnailFrameRect:=Rect(10,10,50,5
0);
```

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property ThumbnailResampleFilter:
TResampleFilter;
```

DESCRIPTION

ThumbnailResampleFilter specifies the filter to use when the application adds a new thumbnail. This enhances the image quality but could slow down the application. Default value is rfFastLinear

Example

```
// insert image 1.jpg and 2.jpg. Only 1.jpg will
be filtered.
ImageEnMView1.ThumbnailResampleFilter:=
rfBSpline;
Idx:=ImageEnMView1.AppendImage;
ImageEnMView1.SetImageFromFile('1.jpg');
ImageEnMView1.ThumbnailResampleFilter:= rfNone;
Idx:=ImageEnMView1.AppendImage;
ImageEnMView1.SetImageFromFile('2.jpg');
```

DECLARATION

```
property ThumbnailsBackground: TColor;
```

DESCRIPTION

ThumbnailsBackground specifies the background color of the thumbnails.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ThumbnailsBackgroundStyle:
TIEBackgroundStyle
```

DESCRIPTION

ThumbnailsBackgroundStyle specifies the thumbnails background style. The thumbnails background region not filled by the image (only when EnableAlphaChannel is true and the image has an alpha channel that makes it transparent).

DECLARATION

```
property ThumbnailsBorderColor: TColor;
```

DESCRIPTION

ThumbnailsBorderColor specifies the color of the thumbnail border.

Example

```
// draw a thin green border to all thumbnails
ImageEnMView1.ThumbnailsBorderWidth:=1;
ImageEnMView1.ThumbnailsBorderColor:=clGreen;
```

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ThumbnailsBorderWidth: integer;
```

DESCRIPTION

ThumbnailsBorderWidth specifies the width of the thumbnail border. Default is 0. This value should be less than ThumbWidth and ThumbHeight. To specify the border color use ThumbnailsBorderColor property.

Example

```
// draw a thin green border to all thumbnails
ImageEnMView1.ThumbnailsBorderWidth:=1;
ImageEnMView1.ThumbnailsBorderColor:=clGreen;
```

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ThumbnailsInternalBorderColor: TColor;
```

DESCRIPTION

ThumbnailsInternalBorderColor specifies the border color when ThumbnailsInternalBorder is true.

DECLARATION

```
property ThumbnailsInternalBorder: Boolean;
```

DESCRIPTION

If ThumbnailsInternalBorder is true a border around thumbnails will be drawn. To specify the color use ThumbnailsInternalBorderColor.

DECLARATION

```
property ThumbsRounded: integer;
```

DESCRIPTION

If ThumbsRounded >0 it specifies that the image corners are rounded. You get maximum round with little values, while large values makes little rounds.

Example

```
ImageEnMView1.ThumbsRounded:=5;
```

ImageEnMView

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property ThumbWidth: integer;
```

DESCRIPTION

ThumbWidth is the width of the thumbnail and of the image. See also ThumbHeight.

DECLARATION

```
property UpperGap: integer;
```

DESCRIPTION

UpperGap is the distance from the image and its border. You can set UpperGap to reserve space for your painting (with OnImageDraw).

Input/output

DECLARATION

```
property EnableAdjustOrientation: Boolean;
```

DESCRIPTION

When this property is true all images which have orientation information (like jpeg with EXIF) will be automatically orientated.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property EnableImageCaching: Boolean;
```

DESCRIPTION

EnableImageCaching allows you to speed up thumbnails painting, saving each drawn image to a cache. This is enabled by default. To disable image caching, set EnableImageCaching to False.

See also
ClearImageCache.

DECLARATION

```
property EnableLoadEXIFThumbnails: Boolean;
```

DESCRIPTION

If true (default) when you request to load thumbnails, ImageEn try to load EXIF thumbnails instead of resampled version of the full image.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property EnableResamplingOnMinor: Boolean;
```

DESCRIPTION

If EnableResamplingOnMinor is true (default), the images are resampled to fit thumbnail size, otherwise the image is resampled only if it is greater than thumbnail size.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure FillFromDirectory(const Directory:
WideString; Limit: integer=-1;
AllowUnknownFormats: Boolean=false; const
ExcludeExtensions: WideString=”;
DetectFileFormat: Boolean=false; const
FilterMask: WideString = “);
```

DESCRIPTION

FillFromDirectory automatically loads all known images inside Directory. FillFromDirectory sets ImageFileName with the image full path and ImageBottomText with the file name.

Parameter	Description
Directory	Directory where to search files.
Limit	Specifies the maximum number of images to load. -1 means no limit.
AllowUnknownFormats	If false (default) loads only known and supported file formats. Otherwise tries to load all files.
ExcludeExtensions	Contains a comma separated list of file extensions to discard (i.e. ‘lyr,all,iev’).
DetectFileFormat	If true then the image type is detected reading the header, otherwise ImageEn looks at filename extension.

Parameter	Description
FilterMask	Contains a comma separated list of file extensions to include. Empty string means “all supported extensions”.

Example

```
ImageEnMView1.Clear;
ImageEnMView1.StoreType := ietThumb;
ImageEnMView1.FillFromDirectory('c:\images');
```

DECLARATION

```
procedure GetImageToFile(idx: integer; const
FileName: WideString);
```

DESCRIPTION

GetImageToFile saves the specified frame (image) to file. The file format is given by the name extension.

Parameter	Description
idx	The image index (0=first image)
FileName	The destination path and file name

Example

```
// separates pages of a multipage tif
ImageEnMView1.MIO.LoadFromFile('multipage.tif');
ImageEnMView1.GetImageToFile(0,'page1.tif');
ImageEnMView1.GetImageToFile(1,'page2.tif');
```

DECLARATION

```
procedure GetImageToStream(idx: integer; Stream:
TStream; ImageFormat: TIOFileType);
```

DESCRIPTION

GetImageToStream saves the specified frame (image) to Stream.

Parameter	Description
Idx	The image index (0=first image)
Stream	The destination stream
ImageFormat	Specifies the output image format (tif,jpeg...)

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ImageCacheSize: integer;
```

DESCRIPTION

ImageCacheSize contains the number of images to be stored in memory instead of a memory mapped file. For example, if you know that TImageEnMView will contain only 20 images then the ImageCacheSize should be 20. The default value is 10.

DECLARATION

```
property ImageCacheUseDisk: Boolean;
```

DESCRIPTION

When ImageCacheUseDisk is true (default) a disk file is used to cache images and view. Otherwise system memory is used. This is useful if you have low disk space or you don't want ImageEn writes on disk. Changing this property has as side effect the call to Clear method.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property JobsRunning: integer;
```

DESCRIPTION

JobsRunning returns the number of threads which currently are loading thumbnails.

Demo
viewers/thumbnails

DECLARATION

```
property JobsWaiting: integer;
```

DESCRIPTION

JobsWaiting returns the number of images/thumbnails which waits to be loaded.

Demo
viewers/thumbnails

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure LoadFromFileOnDemand(const FileName:
WideString);
```

DESCRIPTION

LoadFromFileOnDemand fills ImageFileName property with values like 'FileName::0', 'FileName::1', etc. to load the whole specified multipage file. Frames will be loaded only when they are needed.

Example

```
ImageEnMView1.LoadFromFileOnDemand('input.mpeg');
```

DECLARATION

```
property LoadIconOnUnknownFormat: Boolean;
```

DESCRIPTION

If true, when ImageFileName contains an unknown format ImageEn loads the default file icon (it gets the icon using ShGetFileInfo shellapi function). If false, a question mark is displayed.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
function LoadSnapshot(Stream: TStream):Boolean;
function LoadSnapshot(FileName: WideString):
Boolean;
```

DESCRIPTION

Loads images, caches, texts and thumbnails size from the specified stream or file, saved using SaveSnapshot. This is useful to create caching mechanism like Windows .db files, to load quickly an entire directory of images.

DECLARATION

```
property LookAhead: integer;
```

DESCRIPTION

This property specifies the number of invisible images to load ahead. The default is 0. This property acts only if images are loaded on demand like when you set ImageFileName or FillFromDirectory or LoadFromFileOnDemand.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property MaintainInvisibleImages: integer;
```

DESCRIPTION

This property specifies the number of images to maintain when they are no more visible. The default is 15. Specifying -1 you allow maintaining all images. Setting 0, all no more visible images are discarded. This property acts only if images are loaded on demand like when you set ImageFileName or FillFromDirectory or LoadFromFileOnDemand.

DECLARATION

```
property MIO: TImageEnMIO;
```

DESCRIPTION

The MIO property encapsulates the TImageEnMIO component inside TImageEnMView. The TImageEnMIO component is created the first time you use the MIO property.

Example

```
ImageEnMView1.MIO.LoadFromFile('film.avi');
ImageEnMView1.MIO.Acquire;
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure ReloadImage(imageIndex: integer);
```

DESCRIPTION

This method reloads an image. This works only with on demand image (where you set ImageFileName of ImageID).

DECLARATION

```
property RemoveCorrupted: Boolean;
```

DESCRIPTION

If RemoveCorrupted is True, TImageEnMView automatically removes all corrupted images from the grid.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure SaveSnapshot(Stream: TStream;
SaveCache: Boolean=true; Compressed:
Boolean=false; SaveParams: Boolean = false);
procedure SaveSnapshot(FileName: WideString;
SaveCache: Boolean = true; Compressed:
Boolean=false; SaveParams: Boolean=false);
```

DESCRIPTION

SaveSnapshot saves all images, caches, texts and thumbnails size in the specified stream or file. This is useful to create caching mechanism like Windows .db files, to load quickly an entire directory of images. If SaveCache is true (default) the image caches are saved. This speedup display but require more space. If Compressed is true an LZ compression algorithm is applied. This reduces space but slowdown saving. If SaveParams is true input/output parameters and tags are saved. You can reload a saved snapshot using LoadSnapshot.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
function SetImageFromFile(idx: integer; const
FileName: WideString; SourceImageIndex: integer =
0):Boolean;
```

DESCRIPTION

SetImageFromFile loads an image and assigns it to idx index. SourceImageIndex specifies the source page index when source file is a multipage file (like a TIFF).

Example

```
idx:=ImageEnMView1.AppendImage;
ImageEnMView1.SetImageFromFile(idx,'myfile.jpg');
```

DECLARATION

```
function SetImageFromStream(idx: integer; Stream:
TStream; SourceImageIndex: integer = 0):Boolean;
```

DESCRIPTION

SetImageFromStream loads an image and assigns it to idx index. SourceImageIndex specifies the source page index when source file is a multipage file (like a TIFF).

Example

```
idx:=ImageEnMView1.AppendImage;
ImageEnMView1.SetImageFromStream(idx, stream);
```

DECLARATION

```
property ThreadPoolSize:integer;
```

DESCRIPTION

ThreadPoolSize specifies how many threads can be created to load images. If it is 0 all images are loaded in the main thread.

Selections

DECLARATION

```
procedure BeginSelectImages;
```

DESCRIPTION

Call BeginSelectImages and EndSelectImages when you need to select multiple images without refreshing the component's state. Generally this will speed up the selection process.

Example

```
// select the first 100 images
ImageEnMView1.BeginSelectImages;
for i:=0 to 99 do
ImageEnMView1.SelectedImage:=i;
ImageEnMView1.EndSelectImages;
```

DECLARATION

```
procedure CenterSelected;
```

DESCRIPTION

This method moves ViewX and ViewY to show (centered) the currently selected image.

Example

```
ImageEnMView1.SelectedImage:=10;
ImageEnMView1.CenterSelected;
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property CheckThumbBoundsOnSelect: Boolean;
```

DESCRIPTION

If True ImageEn will check the thumbnail bounding rectangle before select it. The default is False, allowing more flexible selections. It is necessary to keep CheckThumbBoundsOnSelect = false (the default) when using multi-selection.

DECLARATION

```
procedure DeleteSelectedImages;
```

DESCRIPTION

DeleteSelectedImages removes all selected images in one step.

DECLARATION

```
procedure DeSelect;
```

DESCRIPTION

Deselects all selected images.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property EnableMultiSelect: Boolean;
```

DESCRIPTION

EnableMultiSelect enables multi-selections. Default is false (multi-selection disabled).

Example

```
// select images 0 and 1 (you have to set at
design time
ImageEnMView1.EnableMultiSelect:=True)
ImageEnMView1.Deselect;
ImageEnMView1.MultiSelecting:=True;
ImageEnMView1.SelectedImage:=0;
ImageEnMView1.SelectedImage:=1;
ImageEnMView1.MultiSelecting:=False;
```

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure EndSelectImages;
```

DESCRIPTION

Call BeginSelectImages and EndSelectImages when you need to select multiple images without refreshing the component's state. Generally this will speed up the selection process.

Example

```
// select the first 100 images
ImageEnMView1.BeginSelectImages;
for i:=0 to 99 do
ImageEnMView1.SelectedImage:=i;
ImageEnMView1.EndSelectImages;
```

DECLARATION

```
function IsSelected(idx: integer): Boolean;
```

DESCRIPTION

IsSelected returns true if the idx image is currently selected.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure MoveSelectedImagesTo(beforeImage: integer);
```

DESCRIPTION

Moves selected images before the specified image.
To move images after last image set beforeImage=ImageCount.

DECLARATION

```
property MultiSelectedImages[index: integer]: integer;
```

DESCRIPTION

MultiSelectedImages allows you know which images are selected.
Use MultiSelectedImagesCount to know how many images are selected.

Example

```
// replaces all selected images with 'new.jpg'
for i:=0 to ImageEnMView1.MultiSelectedImagesCount-1 do
begin
  selected := ImageEnMView1.MultiSelectedImages[ i ];
  ImageEnMView1.SetImageFromFile( selected , 'new.jpg' );
end;
```

ImageEnMView

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property MultiSelectedImagesCount: integer;
```

DESCRIPTION

MultiSelectedImagesCount returns the number of selected images. Selected indexes are in MultiSelectedImages.

Example

```
// replaces all selected images with "new.jpg"
for i:=0 to ImageEnMView1.MultiSelectedImagesCount-1 do
begin
  selected := ImageEnMView1.MultiSelectedImages[ i ];
  ImageEnMView1.SetImageFromFile( selected , 'new.jpg' );
end;
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property MultiSelecting: Boolean;
```

DESCRIPTION

Set MultiSelecting to True to simulate a CTRL key press. It allows user to select multiple images with mouse or arrow keys without press CTRL key. Also use MultiSelecting to select more than one image using the SelectedImage property. To use multiselections, the EnableMultiSelect property must be true.

Example

```
// select images 0 and 1 (you have to set at
design time
ImageEnMView1.EnableMultiSelect:=True)
ImageEnMView1.Deselect;
ImageEnMView1.MultiSelecting:=True;
ImageEnMView1.SelectedImage:=0;
ImageEnMView1.SelectedImage:=1;
ImageEnMView1.MultiSelecting:=False;
```

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property MultiSelectionOptions:
TIEMultiSelectionOptions;
```

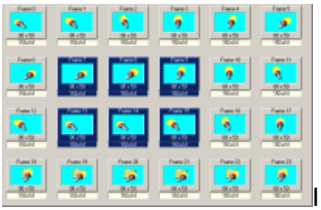
DESCRIPTION

MultiSelectionOptions specifies items selection behavior.

Example

```
// If you do not specify iemoRegion the entire
// row is selected:
ImageEnMView1.MultiSelectionOptions:=[];
```

```
// By specifying iemoRegion only specified
// columns are selected:
ImageEnMView1.MultiSelectionOptions:=[iemoRegion];
```



ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure MultiSelectSortList;
```

DESCRIPTION

This method sorts the selected items list. In this way you have MultiSelectedImages list sorted by image index.

DECLARATION

```
procedure SelectAll;
```

DESCRIPTION

SelectAll selects all images.

DECLARATION

```
property SelectedImage: integer;
```

DESCRIPTION

SelectedImage gets/sets the currently selected image. The selected image is shown with a contour. You can get the bitmap (TBitmap object) of selected image with Bitmap property.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property SelectionAntialiased: Boolean;
```

DESCRIPTION

When true the selection is antialiased.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property SelectionColor: TColor;
```

DESCRIPTION

SelectionColor is the color of the selection.

Example

```
ImageEnMView1.SelectionStyle:=iessAround;
```


```
ImageEnMView1.SelectionStyle:=iessACD;
```

Unit ImageEnMView

TImageEnMView

DECLARATION

```
TIESStyle = (iessAround, iessACD);
```

DESCRIPTION

Value	Description
iessAround	Draw a rectangle around the thumbnail.
iessACD	Use HighlightColor for background and HighlightTextColor for text (default).

DECLARATION

```
property SelectionStyle: TIESStyle;
```

DESCRIPTION

SelectionStyle specifies how a selected thumbnail will be shown.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property SelectionWidth: integer;
```

DESCRIPTION

SelectionWidth is the width of the selection.

This value must be less than HorizBorder and VertBorder.

DECLARATION

```
property SelectionWidthNoFocus: integer;
```

DESCRIPTION

This property specifies the selection width when the component hasn't the focus. The default value is 1.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure SelectSeek(pos: TIESeek);
```

DESCRIPTION

SelectSeek moves the current selected image at the pos position. SelectSeek scrolls to make the selected image visible.

Example

```
// This code loads "film.avi" and select the
// first image
ImageEnMView1.MIO.LoadFromFile('film.avi');
ImageEnMView1.SelectSeek(iskFirst);
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property TrackMouseSelection: Boolean;
```

DESCRIPTION

If TrackMouseSelection is true a semi-transparent rectangle is show during mouse selection. Default is False.

Demo
multipage\multi

DECLARATION

```
procedure UnSelectImage(idx: integer);
```

DESCRIPTION

UnSelectImage unselects a specified image when multiselection is enabled.

DECLARATION

```
property VisibleSelection: Boolean;
```

DESCRIPTION

If VisibleSelection is true, then show current selection.

ImageEnMView

Unit ImageEnMView

TImageEnMView

User Interaction

Declaration

```
function GetImageVisibility(idx:
integer):integer;
```

Description

GetImageVisibility returns 0 when idx image is invisible, 1 when is partially visible or 2 when is fully visible.

Declaration

```
property HScrollBarParams: TIEScrollBarParams;
```

Description

The HScrollBarParams property allows an application to customize the horizontal scroll bar behavior like tracking (display refresh on mouse dragging), left/right buttons pixel scroll, pageleft/pageright pixel scroll.

Example

```
// disable tracking
ImageEnMView1.HScrollBarParams.Tracking:=False;
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
function ImageAtGridPos(row, col: integer): integer;
```

DESCRIPTION

ImageAtGridPos returns the index of the image at row, col position. The top-left image is row=0 and col=0.

DECLARATION

```
function ImageAtPos(x, y: integer; checkBounds: boolean = true): integer;
```

DESCRIPTION

Use ImageAtPos when you want to know which image is at the specified location within the control. The x, y parameters specify the position in "client coordinates". If checkBounds is True, thumbnail bounds are checked (False, is useful to find "near to" thumbnail). If there is no thumbnail at the specified position, ImageAtPos returns -1.

ImageEnMView

Example

```
// Display in the status bar file name of the
image where is the mouse
procedure TForm1.ImageEnMView1MouseMove(Sender:
TObject; Shift: TShiftState; X, Y: Integer);
begin
  StatusBar1.SimpleText :=
ImageEnMView1.ImageFileName[
ImageEnMView1.ImageAtPos(x,y)];
end;
```

DESCRIPTION

Allows an application to customize the scroll bars behavior like tracking (display refresh on mouse dragging), up/down buttons pixel scroll, pagedown/up pixel scroll.

Properties

DECLARATION

```
property LineStep: integer;
```

DESCRIPTION

LineStep is the number of pixels to scroll when the user clicks on UP or DOWN arrows. Setting this property to -1, the component scrolls by one thumbnail at a time. Default value is -1.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property PageStep: integer;
```

DESCRIPTION

PageStep is the number of pixels to scroll when the user clicks near the cursor (PAGEUP or PAGEDOWN). Setting this property to -1, the component scrolls by one page (client height). Default value is -1.

DECLARATION

```
property Tracking: Boolean
```

DESCRIPTION

Set Tracking to False to disable display refreshing during mouse dragging. Default value is True.

DECLARATION

```
function ImageAtGridPos(row, col:
integer):integer;
```

DESCRIPTION

ImageAtGridPos returns the index of the image at row, col position. Row = 0 and col=0 specifies top-left image.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
function ImageAtPos(x, y: integer; checkBounds:
Boolean = true):integer;
```

DESCRIPTION

Use ImageAtPos when you want to know which image is at the specified location within the control. x, y parameters specify the position in “client coordinates”. If checkBounds is True, thumbnail bounds are checked (False, is useful to find “near to” thumbnail). If there is no thumbnail at the specified position, ImageAtPos returns -1.

Example

```
// Display in the status bar file name of the
image where is the mouse
procedure TForm1.ImageEnMView1MouseMove(Sender:
TObject; Shift: TShiftState; X, Y: Integer);
begin
  StatusBar1.SimpleText :=
    ImageEnMView1.ImageFileName[
      ImageEnMView1.ImageAtPos(x,y)];
end;
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
function InsertingPoint(x, y: integer): integer;
```

DESCRIPTION

The InsertingPoint function returns the index of the image before or after the x, y position. It is useful when an image needs to be inserted at a particular x, y mouse position.

Example

```
// this drag/drop event copy all selected images
// of ImageEnMView1 to ImageEnMView2, starting at
// X,Y mouse position
procedure TForm1.ImageEnMView2DragDrop(Sender,
Source: TObject; X, Y: Integer);
var
i: integer;
idx,im: integer;
iBmp: TBitmap;
begin
  im := ImageEnMView2.InsertingPoint(X,Y);
  for i:=0 to
    ImageEnMView1.MultiSelectedImagesCount-1 do
    begin
      idx := ImageEnMView1.MultiSelectedImages[i];
      iBmp:=ImageEnMView1.GetBitmap( idx );
      ImageEnMView2.InsertImage(im);
      ImageEnMView2.SetImage(im, iBmp);
     inc(im);
     ImageEnMView1.ReleaseBitmap( idx );
  end;
end;
```

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
function IsVisible(idx: integer): Boolean;
```

DESCRIPTION

IsVisible returns true if idx image is currently visible.

DECLARATION

```
property KeyInteract: TIEMKeyInteract
```

DESCRIPTION

KeyInteract sets which keyboard activities TImageEnMView handles automatically.

DECLARATION

```
property MouseInteract: TIEMMouseInteract;
```

DESCRIPTION

MouseInteract specifies the mouse interactions handled by TImageEnMView.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ScrollBars: TScrollType;
```

DESCRIPTION

ScrollBars determines whether the TImageEnView control has any scroll bars. If the component does not need scrollbars they are not shown.

Value	Description
ssNone	The control has no scroll bars.
ssHorizontal	The control has a single scroll bar on the bottom edge when needed.
ssVertical	The control has a single scroll bar on the right edge when needed.
ssBoth	The control has a scroll bar on both the bottom and right edges when needed.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property ScrollBarsAlwaysVisible: Boolean;
```

DESCRIPTION

When the ScrollBarsAlwaysVisible property is true, the scroll bars specified in ScrollBars property will be displayed, even if this is not necessary.

DECLARATION

```
property VScrollBarParams: TIEScrollBarParams;
```

DESCRIPTION

The VScrollBarParams property allows an application to customize the vertical scroll bar behavior like tracking (display refresh on mouse dragging), up/down buttons pixel scroll, pagedown/up pixel scroll.

Example

```
// disable tracking
ImageEnMView1.VScrollBarParams.Tracking:=False;
```

ImageEnMView

Unit ImageEnMView
TImageEnMView

Animations and transitions

Declaration

```
property ImageDelayTime[idx: integer]: integer;
```

Description

ImageDelayTime is the time in milliseconds the image idx will be shown on playing (Playing property).

Declaration

```
property Playing: Boolean;
```

Description

Set Playing to True to select images consecutively each for ImageDelayTime time. During play, DisplayMode property is set to mdSingle and the Deselect method is called.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property PlayLoop: Boolean;
```

DESCRIPTION

Set PlayLoop to true to continuously loop playing.

DECLARATION

```
property TransitionDuration: integer;
```

DESCRIPTION

TransitionDuration specifies the duration of the transition in milliseconds.

Example

```
// design time properties...
ImageEnMView1.DisplayMode:=mdSingle;
ImageEnMView1.TransitionEffect:=iettCrossDissolve;
ImageEnMView1.TransitionDuration:=1500;
// display next frame using cross dissolve
ImageEnMView1.VisibleFrame:=ImageEnMView1.VisibleFrame+1;
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property TransitionEffect: TIETransitionType;
```

DESCRIPTION

TransitionEffect specifies the effect to apply when the application changes the currently displayed frame. The DisplayMode must be mdSingle. To change current frame, use the VisibleFrame property.

Example

```
// design time properties...
ImageEnMView1.DisplayMode:=mdSingle;
ImageEnMView1.TransitionEffect:=iettCrossDissolve;
ImageEnMView1.TransitionDuration:=1500;
// display next frame using cross dissolve
ImageEnMView1.VisibleFrame:=ImageEnMView1.VisibleFram
e+1;
```

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property TransitionRunning: Boolean;
```

DESCRIPTION

TransitionRunning is true whenever a transition is running.

Example

```
// design time properties...
ImageEnMView1.DisplayMode:=mdSingle;
ImageEnMView1.TransitionEffect:=iettCrossDissolve;
ImageEnMView1.TransitionDuration:=1500;
// display next frame using cross dissolve
ImageEnMView1.VisibleFrame:=ImageEnMView1.VisibleFrame+1;
// wait transition end
While ImageEnMView1.TransitionRunning do
Application.ProcessMessages;
ShowMessage('transition done!');
```

Others

DECLARATION

```
function CalcGridWidth():integer;
```

DESCRIPTION

CalcGridWidth calculates the grid width (the number of columns of the grid). Useful when GridWidth is -1 (automatic).

DECLARATION

```
function GetLastOp: integer;
```

DESCRIPTION

GetLastOp is undocumented. Please don't use.

DECLARATION

```
function GetLastOpIdx: integer;
```

DESCRIPTION

GetLastOpIdx is undocumented. Please don't use.

Unit ImageEnMView
TImageEnMView

Declaration

```
procedure IEBeginDrag(Immediate: Boolean;
Threshold: Integer = -1);
```

Description

This method must be used in replacement of BeginDrag, when you need to perform a drag & drop.

Demo
dragdrop\timageenmview_dragdrop

Declaration

```
procedure IEEndDrag;
```

Description

This method must be used in replacement of EndDrag, when you need to perform a drag & drop.

Declaration

```
property ImageEnVersion:string;
```

Description

This is a published property which returns the ImageEn version as string.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property Proc: TImageEnProc;
```

DESCRIPTION

The Proc property encapsulates the TImageEnProc component inside TImageEnMView. The TImageEnProc component is created the first time you use Proc property.

Example

```
ImageEnMView1.Proc.Negative;
// negate selected image
```

DECLARATION

```
property StoreType: TIEStoreType;
```

DESCRIPTION

StoreType specifies how an image is stored in the TImageEnMView component.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
procedure Update;
```

DESCRIPTION

Updates display with the actual content and properties of the component.

DECLARATION

```
procedure UpdateCoords;
```

DESCRIPTION

For internal use only.

Events

DECLARATION

```
property OnAllDisplayed: TNotifyEvent;
```

DESCRIPTION

This event occurs when all images are loaded and displayed.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property OnBeforeImageDrawEx:
TIEImageDrawEventEx;
```

DESCRIPTION

OnBeforeImageDrawEx occurs just before the image idx is painted. Left, Top are the left-top coordinates where the image will be drawn (Dest bitmap coordinates). Dest is the destination bitmap (the canvas where the image must be drawn). ThumbRect specifies the destination rectangle for the image. You can change this parameter, so the image will be drawn where you want.

When this event is assigned, the selection is not shown.

Look at 'customthumbs' demo for more info.

DECLARATION

```
property OnBeforeImageDraw: TIEImageDrawEvent;
```

DESCRIPTION

OnBeforeImageDraw event occurs just before an image is drawn. It is useful to prepare some parameters prior to drawing the image (as ImageTopText, ImageBottomText and ImageInfoText).

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property OnDrawProgress: TIEMProgressEvent;
```

DESCRIPTION

The OnDrawProgress event is called on paint for each image drawn.

Example

```
procedure TForm1.ImageEnMView1DrawProgress(Sender:
TObject; per, idx: Integer);
begin
  ProgressBar1.Position := per;
end;
```

DECLARATION

```
property OnFinishWork: TNotifyEvent;
```

DESCRIPTION

This event occurs whenever an image processing task or input/output terminates. This event is called after the last call to OnProgress, so you can reset here your progress bar.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property OnImageAtPos: TIEImageAtPosEvent;
```

DESCRIPTION

Event called whenever it is necessary to check if specified coordinates are inside a thumbnail.

DECLARATION

```
property OnImageDeselect: TIEImageSelectEvent;
```

DESCRIPTION

This event occurs when an image is deselected by an user action. EnableMultiSelect must be true.

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property OnImageDraw: TIEImageDrawEvent;
```

DESCRIPTION

This event is called whenever an image is painted.

Example

```
// This method display the image index and sizes on bottom
// of the thumbnail.
// Set properly the BottomGap property
procedure TForm1.ImageEnMView1ImageDraw(Sender: TObject;
idx: Integer;
Left, Top: Integer; Canvas: TCanvas);
begin
  with canvas do
  begin
    Font.Height:=15;
    Font.Color:=clWhite;
    Textout(Left,Top+imageenmview1.ThumbHeight-
       imageenmview1.bottomgap+2,inttostr(idx));
    Textout(Left,Top,
      IntToStr(imageenmview1.imageWidth[idx]) +
        'x' + IntToStr(imageenmview1.imageHeight[idx]));
  end;
end;
```

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property OnImageDraw2: TIEImageDraw2Event;
```

DESCRIPTION

This event is called whenever an image is painted. The thumbnail rectangle is available.

DECLARATION

```
property OnImageIDRequestEx:
TIEImageIDRequestExEvent;
```

DESCRIPTION

The OnImageIDRequestEx event is called if you have inserted a value in the ImageID property and the image must be shown.

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property OnImageIDRequest:
TIEImageIDRequestEvent;
```

DESCRIPTION

This event is called if you have inserted a value in ImageID property and the image must be shown.

ID is the ID you have specified in ImageID property. Bitmap is the bitmap to show. The bitmap is copied in TImageEnMView, and must be freed when it is no longer necessary.

DECLARATION

```
property OnImageSelect: TIEImageSelectEvent;
```

DESCRIPTION

This event is called whenever an image is selected.

DECLARATION

```
property OnIOProgress: TIEProgressEvent;
```

DESCRIPTION

OnIOProgress event is called on input/output operations.

ImageEnMView

Unit ImageEnMView
TImageEnMView

DECLARATION

```
property OnMouseEnter: TNotifyEvent;
```

DESCRIPTION

OnMouseEnter occurs when the mouse pointer moves over the component. Write OnMouseEnter event handler to take specific action when the user moves the mouse over the component.

DECLARATION

```
property OnMouseLeave: TNotifyEvent;
```

DESCRIPTION

OnMouseLeave occurs when the mouse pointer moves off of the component. Write OnMouseLeave event handler to take specific action when the user moves the mouse off the component.

DECLARATION

```
property OnPlayFrame: TIEPlayFrameEvent;
```

DESCRIPTION

This event occurs whenever a frame is played. Look also Playing.

Unit ImageEnMView

TImageEnMView

DECLARATION

```
property OnViewChange: TViewChangeEvent;
```

DESCRIPTION

OnViewChange notifies when ViewX or ViewY properties change.

DECLARATION

```
property OnWrongImage: TIEWrongImageEvent;

TIEWrongImageEvent = procedure(Sender: TObject;
OutBitmap: TIEBitmap; idx: integer; var Handled:
Boolean) of object;
```

DESCRIPTION

OnWrongImage occurs whenever TImageEnMView cannot load the image specified in ImageFileName property, for instance when the file is corrupted or not recognized.

Applications can specify an alternative bitmap to display by changing the OutBitmap property.

Example

```
procedure Form1_OnWrongImage(Sender: TObject;
OutBitmap: TIEBitmap; idx: integer; var Handled:
Boolean);
var
io: TImageEnIO;
begin
  io := TImageEnIO.CreateFromBitmap(OutBitmap);
  io.LoadFromFile('error_image.bmp');
  io.Free;
  Handled:=True;
end;
```

Chapter 5

TImageEnVect

Chapter 5. ImageEnVect

TImageEnVect inherits from TImageEnView (has all method and properties), and handles vectorial objects and measurements.

Like TImageEnView, TImageEnVect can be linked to TImageEnProc and TImageEnIO, for image processing and input/output of the background image. You can also use embedded objects in IO and Proc properties, without use further TImageEnIO and TImageEnProc components. TImageEnVect has methods to load and save vectorial objects (doesn't need TImageEnIO for vectorial load/save).

Unit ImageEnVect

TImageEnVect

TImageEnVect can display and allow editing of vectorial objects which are displayed on top of a background image. Vectorial objects may be lines, boxes ellipses, image, text, rulers, angles, polylines and memos. Objects are selectable, moveable and resizeable, have transparency, pen and brush color, pen and brush styles, thickness, and shadows. Objects can be saved and loaded from a file and the background image and its objects may be reloaded from a file or a stream. Objects can be merged into the background image as well; however, merged objects are no longer editable. TImageEnVect is very versatile because it uses the same fileIO and image processing functions available to TImageEnView via the encapsulated ImageEnIO and ImageEnProc components.

Methods and Properties

Display

AllObjectsHidden	BitmapResampleFilter	ObjAntialias
ObjectsExtents	ObjEnableFastDrawing	ObjGraphicRender
ZoomObjectsWidth		

User Interaction

AllowOutOfBitmapMoving	AllowOutOfBitmapPolylines
CancelInteracts	CenterNewObjects
EnableRangeObjectsSelection	FindObjectAt
IsEditMode	MaxMovingDistance

Unit ImageEnVect
TImageEnVect

MouseInteractVt
ObjEditOnNewText
PolylineEndingMode
SelectOnMouseDown
UseCentralGrip

Measurement

FloatDigits
FloatPrecision
GetAngleValue
GetObjDiagLen
GetPolylineArea
GetPolylineCentroid
GetPolylineLen
GetSelectionArea
GetSelectionCentroid
GetSelectionLen
MeasureCoefX
MeasureCoefY
MeasureHintBorder1
MeasureHintBorder2
MeasureHintBrush
MeasureHintFont
MeasureTrack
MUnit
ScaleFactor
SetScaleFromPixels
SetScaleFromSelectionLen

Object Editing

AddNewObject
CreateImageFromSelectedArea
CreatePolygonFromEdge
CreatePolygonsFromSelection
GetIndexFromObj
GetMemoFormattedString

Unit ImageEnVect
TImageEnVect

GetObjFromID
GetObjFromIndex
GetObjFromName
GetObjRect
MemoEdit
MemoEditingGetCharInfo
MemoEditingSetCharInfo
ObjSetTBitmap
ObjTextEditMode
RemoveAllObjects
RemoveObject
RemovePolygonJaggedEdges
RotateAllObjects
RotateObject
SetObjBackTo
SetObjBitmapICO
SetObjFont
SetObjFrontOf
SetObjPolylinePoints
SetObjRect
SetObjTextCurve
SetObjTextCurveFromPolyline
SetObjTextCurveShape
SimplifyPolygon

Rendering And Copying

CopyAllObjectsTo
CopyObjectsToBack
CopyObjectToBack
CopyObjectTo
DrawObjectsToBitmap
DrawOneObjectToBitmap

Input/output

ImportDXF
LoadFromFileAll
LoadFromFileIEV
LoadFromStreamAll
LoadFromStreamIEV
LoadObjectsFromTIFF

Unit ImageEnVect
TImageEnVect

SaveObjectsToTIFF	SaveToFileAll	SaveToFileIEV
SaveToStreamAll	SaveToStreamIEV	SelAllObjects
SetObjBitmapFromFile	SetObjBitmapFromStrea	

Clipboard

ObjCopyToClipboard	ObjCutToClipboard	ObjIsClipboardAvailable
ObjPasteFromClipboard		

Object Properties

ObjAngleShowSmall	ObjArcEndingAngle	ObjArcStartingAngle
ObjAspectRatio	ObjBeginShape	ObjBitmapAlpha
ObjBitmapBorder	ObjBitmap	ObjBlendOperation
ObjBoxHighLight	ObjBoxInnerSelectable	ObjBrushColor
ObjBrushStyle	ObjEndShape	ObjFontAngle
ObjFontHeight	ObjFontLocked	ObjFontName
ObjFontStyles	ObjFontQuality	ObjHeight
ObjID	ObjIsVisible	ObjKind
ObjLabelBorder	ObjLabelBrushColor	ObjLabelBrushStyle
ObjLabelPosition	ObjLeft	ObjMemoBorderColor

Unit ImageEnVect

TImageEnVect

Object Properties

ObjMemoBorderStyle	ObjMemoCharsBrushStyle	ObjMemoFixedHeight
ObjMemoHasBitmap	ObjMemoLineSpace	ObjMemoMarginBottom
ObjMemoMarginLeft	ObjMemoMarginRight	ObjMemoMarginTop
ObjName	ObjPenColor	ObjPenStyle
ObjPenWidth	ObjPolylineClosed	ObjPolylinePointsCount
ObjPolylinePoints	ObjRulerQuoteHorizon	ObjRulerType
ObjRulerUnit	ObjShapeHeight	ObjShapeWidth
ObjSoftShadow	ObjStyle	ObjTextAlign
ObjTextAutoSize	ObjTextCurveCharRot	ObjTextEditable
ObjText	ObjTop	ObjTransparency
ObjUserDataLength	ObjUserData	ObjWidth

Grips

ObjGripBrush	ObjGripImage
ObjGripPen	ObjGripShape
ObjGripSize	

Selection

AddSelObject	CopySelectedObjectsTo
IsSelObject	MaxSelectionDistance

Unit ImageEnVect
TImageEnVect

Object Properties

SelObjects
SelObjectsCount
UnSelAllObjects
UnSelObject

Undo/Redo

ObjAutoUndo
ObjCanUndo
ObjClearAllUndo
ObjClearUndo
ObjSaveUndo
ObjUndo
ObjUndoAt
ObjUndoCount
ObjUndoLimit
ObjUndoMode

Others

InsertingPen
ObjectsCount
ShareBitmaps

Events

OnActivateTextEdit
OnAfterDrawObject
OnBeforeDrawObject
OnBeforeVectorialChanged
OnDeactivateTextEdit
OnDragLenEnd
OnMeasureHint
OnNewObject
OnObjectClick
OnObjectDblClick
OnObjectMoveResize
OnObjectOver

Unit ImageEnVect

TImageEnVect

Object Properties

OnPresentMeasure

OnSelectObject

OnTextEditCursorMoved

OnTextKeyDown

OnUserDeselectObject

OnUserSelectObject

OnVectorialChanged

Display

DECLARATION

```
property AllObjectsHidden: Boolean;
```

DESCRIPTION

Set AllObjectsHidden to hide/show all objects at the same time.

DECLARATION

```
property BitmapResampleFilter: TResampleFilter;
```

DESCRIPTION

BitmapResampleFilter sets the filter to apply to all bitmap (iekBitmap) objects.

DECLARATION

```
property ObjAntialias: Boolean;
```

DESCRIPTION

If ObjAntialias is true then ImageEn draws objects with an anti-alias filter. This will slow down the drawing but enhances the quality.

DECLARATION

```
property ObjectsExtents: TRect;
```

DESCRIPTION

ObjectsExtents is the bounding rectangle of all vectorial objects, expressed in bitmap coordinate.

Read-Only

DECLARATION

```
property ObjEnableFastDrawing: Boolean;
```

DESCRIPTION

When true (default), moving objects or doing other visual operation disables antialiasing and shadows. If you have a fast machine, disable this property to increase user graphical experience.

DECLARATION

```
property ObjGraphicRender: Boolean;
```

DESCRIPTION

Enables antialiasing and alpha channel operations when GDIPlus is not installed.

DECLARATION

```
property ZoomObjectsWidth: Boolean;
```

DESCRIPTION

Set ZoomObjectsWidth to apply zoom to vectorial objects (lines, boxes, etc.) .

User Interaction

DECLARATION

```
property AllowOutOfBitmapMoving: Boolean;
```

DESCRIPTION

If AllowOutOfBitmapMoving is true (default) objects can go out of background bitmap.

DECLARATION

```
property AllowOutOfBitmapPolylines: Boolean;
```

DESCRIPTION

If AllowOutOfBitmapPolylines is true then polylines can be painted out of background bitmap bounding box.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure CancelInteracts;
```

DESCRIPTION

CancelInteracts cancels all current mouse interaction (inserting or modifying objects).

DECLARATION

```
property CenterNewObjects: Boolean;
```

DESCRIPTION

When user inserts new objects with a single click, this property controls if the new object is centered at mouse position (True) or if its top-left side is on the mouse position (False - default).

DECLARATION

```
property EnableRangeObjectsSelection: Boolean;
```

DESCRIPTION

If True (default) you can select multiple objects dragging a rectangle with the mouse.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
function FindObjectAt(x,y: integer; var Distance: double):integer;
```

DESCRIPTION

FindObjectAt gets the object near the client coordinates x, y. Distance will contain the object's distance from the specified coordinates. Return a value <0 if no object is found.

See Also

MaxSelectionDistance

Example

```
// hobj will contain the object at 100,100.
Hobj := FindObjectAt(100,100,distance);
```

ImageEnVect

Unit ImageEnVect

TImageEnVect

DECLARATION

```
function IsEditMode: Boolean;
```

DESCRIPTION

IsEditMode returns true when TImageEnVect is editing a TEXT or MEMO object.

DECLARATION

```
property MaxMovingDistance: integer;
```

DESCRIPTION

MaxMovingDistance represents the maximum pointer distance from the object (in pixels) that allows object moving. Default is 1.

Example

```
ImageEnVect1.MaxMovingDistance := 3;
// 3 pixels around
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property MouseInteractVt: TIEMouseInteractVt;
```

DESCRIPTION

MouseInteractVt selects which mouse actions TImageEnVect will handle automatically.

Example

```
//To allow moving and resizing objects
ImageEnVect1.MouseInteractVt:=[miObjectSelect];
```

DECLARATION

```
property ObjEditOnNewText: Boolean;
```

DESCRIPTION

When user inserts a new text object ImageEn begins editing on it. If you don't want this, set ObjEditOnNewText to false. This will allow inserting text objects without automatic editing.

DECLARATION

```
property PolylineEndingMode:
TIEPolylineEndingMode;
```

DESCRIPTION

Specifies how terminate a polyline. Default is ieemDoubleClick. Using ieemManual you have to interrupt manually the inserting call CancelInteracts or setting a new value in MouseInteractVt.

Example

```
// terminate a polyline without double-clicking
ImageEnVect.PolylineEndingMode:=ieemMouseUp;
```

DECLARATION

```
property SelectOnMouseDown: Boolean;
```

DESCRIPTION

If SelectOnMouseDown is True (default from version 2.1.5), then a mouse down action selects an object, otherwise it waits mouse up to select.

DECLARATION

```
property UseCentralGrip: Boolean;
```

DESCRIPTION

If UseCentralGrip is true (default is False), allows the user to move an object only by using the central grip. If false, it allows the user to move an object by just clicking on it (the central grip disappears).

Measurement

DECLARATION

```
property FloatDigits: integer;
```

DESCRIPTION

FloatDigits is the number of decimal digits will be shown on measurement tasks (distances and areas).

Unit ImageEnVect
TImageEnVect

DECLARATION

```
property FloatPrecision: integer;
```

DESCRIPTION

FloatPrecision is the total number of digits (including decimals) that are shown on measurement tasks (distances and areas).

DECLARATION

```
function GetAngleValue(hobj: integer): double;
```

DESCRIPTION

GetAngleValue returns the measured angle in degrees for an iekAngle object.

Unit ImageEnVect
TImageEnVect

DECLARATION

```
function GetObjDiagLen(hobj: integer): double;
```

DESCRIPTION

GetObjDiagLen returns the diagonal length of the specified object. For a box (image or rectangle) it is the diagonal length. For a line it is the line or ruler length. Ruler calculates distance differently, so you should use this routine instead:

```
function FixedGetObjDiagLen(ie: TImageEnVect; hobj:
integer): double;
var
  lx, ly: double;
  r: TRect;
begin
  ie.GetObjRect(hobj, r);
  lx := abs(r.Right-r.Left) * ie.MeasureCoefX;
  ly := abs(r.Bottom-r.Top) * ie.MeasureCoefY;
  result := sqrt(lx * lx + ly * ly);
end;
```

DECLARATION

```
function GetPolylineArea(hobj: integer): double;
```

DESCRIPTION

GetPolylineArea returns the area of the specified polyline. It closes the polyline if necessary.

DECLARATION

```
function GetPolylineCentroid(hobj:
integer):TPoint;
```

DESCRIPTION

GetPolylineCentroid calculates x, y centroid coordinates of the specified polyline. The point is in bitmap coordinates.

Unit ImageEnVect
TImageEnVect

DECLARATION

```
function GetPolylineLen(hobj: integer): double;
```

DESCRIPTION

GetPolylineLen returns the length (perimeter) of the specified polyline.

If the polyline is composed of two points, GetPolylineLen calculates the line length. If the polyline is composed of three or more points, GetPolylineLen calculates the perimeter.

DECLARATION

```
function GetSelectionArea: double;
```

DESCRIPTION

GetSelectionArea returns the area of current selection.

DECLARATION

```
function GetSelectionCentroid: TPoint;
```

DESCRIPTION

GetSelectionCentroid calculates x, y coordinates of the selection's centroid. The point is in bitmap coordinates.

DECLARATION

```
function GetSelectionLen: double;
```

DESCRIPTION

GetSelectionLen returns the length (perimeter) of current selection. If the selection is composed of only two points, GetSelectionLen calculates the line length; if the selection is composed of three or more points, GetSelectionLen calculates the perimeter.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property MeasureCoefX: double;
```

DESCRIPTION

MeasureCoefX specifies a coefficient to convert a bitmap length to a real world length.

Example

```
// convert 10 horizontal pixels to the specified
// measure unit (MUnit).
Real_length := ImageEnVect.MeasureCoefX * 10;
```

DECLARATION

```
property MeasureCoefY: double;
```

DESCRIPTION

MeasureCoefY specifies a coefficient to convert a bitmap length to a real world length.

Example

```
// convert 10 vertical pixels to the specified
// measure unit (MUnit).
Real_length := ImageEnVect.MeasureCoefY * 10;
```

DECLARATION

```
property MeasureHintBorder1: TColor;
```

DESCRIPTION

MeasureHintBorder1 specifies the border color used for measures hint. It is used to draw the top-left sides.

DECLARATION

```
property MeasureHintBorder2: TColor;
```

DESCRIPTION

MeasureHintBorder2 specifies the border color used for measures hint. It is used to draw the bottom-right sides.

DECLARATION

```
property MeasureHintBrush: TBrush;
```

DESCRIPTION

MeasureHintBrush specifies the brush used for measures hint. It is used to draw the hint background.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property MeasureHintFont: TFont;
```

DESCRIPTION

MeasureHintFont specifies the font used for measures hint.

DECLARATION

```
property MeasureTrack: Boolean;
```

DESCRIPTION

If MeasureTrack is true, on measurement tasks will be calculated as the mouse moves.

DECLARATION

```
property MUnit: TIEUnits;
```

DESCRIPTION

MUnit specifies the measurement unit used for displaying measurement tasks and for the return values of GetSelectionArea and GetSelectionLen methods.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ScaleFactor: double;
```

DESCRIPTION

ScaleFactor specifies the scale factor (default 1).

In the common representation X:Y (ex. 1:100000) is the Y value (100000). This value, with DpiX and DpiY (and measure unit MUnit), weight the measurements of areas and line lengths.

Example

```
// Sets a scale factor of 1:100000
ImageEnVect1.ScaleFactor := 100000;
```

Unit ImageEnVect

TImageEnVect

Declaration

```
procedure SetScaleFromPixels(px: integer; mm:
double);
```

Description

SetScaleFromPixels changes the ScaleFactor property such that px pixels correspond to one mm (MUnit).

Example

```
// This code makes so that 100 pixels are equals
// to 1 meter.
ImageEnVect1.MUnit := ieuMETERS;
ImageEnVect1.SetScaleFromPixels(100,1);
```

Declaration

```
procedure SetScaleFromSelectionLen(mm: double);
```

Description

SetScaleFromSelectionLen makes the perimeter of the selection region correspond to one mm (MUnit) by changing ScaleFactor property.

Unit ImageEnVect

TImageEnVect

Object Editing

DECLARATION

```
function AddNewObject: integer;
function AddNewObject(Kind: TIEVObjectKind; Rect:
TRect; Color: TColor):integer; overload;
```

DESCRIPTION

AddNewObject inserts a new object and returns a handle to the object. The second overload allows specifying object Kind, rectangle and pen color.

Example

Unit ImageEnVect

TImageEnVect

Following three blocks of code produces the same result:

```
1)
h := ImageEnVect.AddNewObject;
ImageEnVect.ObjKind[h] := iekBOX;
ImageEnVect.SetObjRect(h, Rect(10,10,100,100));
ImageEnVect.ObjPenColor[h] := clRed;

2)
ImageEnVect.ObjKind[-1] := iekBOX;
ImageEnVect.SetObjRect(-1, Rect(10,10,100,100));
ImageEnVect.ObjPenColor[-1] := clRed;
ImageEnVect.AddNewObject;

3)
ImageEnVect.AddNewObject(iekBOX,
Rect(10,10,100,100), clRed);
```

ImageEnVect

Example

```
// this example paints a red line from 10, 10
// inside a rectangle of 100,100
with ImageEnVect1 do
begin
  hobj := AddNewObject; // hobj is an integer
  ObjKind[hobj] := iekLINE;
  ObjLeft[hobj] := 10;
  ObjTop[hobj] := 10;
  ObjWidth[hobj] := 100;
  ObjHeight[hobj] := 100;
  ObjPenColor[hobj] := clRed;
end;

// this example paints a red line from 10,10 inside
// a rectangle of 100,100
// The -1 index is the next image to create
with ImageEnVect1 do
begin
  ObjKind[-1] := iekLINE;
  ObjLeft[-1] := 10;
  ObjTop[-1] := 10;
  ObjWidth[-1] := 100;
  ObjHeight[-1] := 100;
  ObjPenColor[-1] := clRed;
  AddNewObject;
end;
```

```
// this example paint a red line from 10,10 inside
// a rectangle of 100,100
// The -2 index is the latest image created
with ImageEnVect1 do
begin
  AddNewObject;
 // first create the object, then set its
 // properties
  ObjKind[-2] := iekLINE;
  ObjLeft[-2] := 10;
  ObjTop[-2] := 10;
  ObjWidth[-2] := 100;
  ObjHeight[-2] := 100;
  ObjPenColor[-2] := clRed;
end;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
function CreateImageFromSelectedArea(feather: integer=0; smooth: Boolean=false):integer;
```

DESCRIPTION

CreateImageFromSelectedArea creates an image object (iekBitmap) from the selected area of background image.

The feather value (0 - 255) is how much feathering you want the object to have. The feather function creates a gradient around the object but in the alpha channel so it gets a smoother look by smoothing the edges. The smooth value tells if an anti-jagging filter should be applied to the alpha channel. If you use the feathering function, you should set smooth to true.

Example

```
hobj := ImageEnVect1.CreateImageFromSelectedArea;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
function CreatePolygonFromEdge(x,y: integer;
maxfilter: Boolean; tolerance: integer): integer;
```

DESCRIPTION

CreatePolygonFromEdge creates a closed polyline (polygon) making a flood fill starting from x, y point.

Set maxfilter to True to apply a maximum filter that removes noise. Tolerance specifies the color difference between starting pixel and testing pixel. CreatePolygonFromEdge returns the object index just created. The kind of the new object is iekPOLYLINE.

This method is useful to create a polygon following the image's edges.

Example

```
// create a new polygon when user presses left
// mouse button over the image
procedure TForm1.ImageEnVect1MouseDown(Sender:
TObject; Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
var
hobj: integer;
begin
  hobj :=
      ImageEnVect1.CreatePolygonFromEdge(X,Y,
        true,25);
  ImageEnVect1.ObjPenColor[hobj] := clRed;
end;
```

ImageEnVect

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure CreatePolygonsFromSelection;
```

DESCRIPTION

Create polygons from current selection. This method creates the right number of polygons which composes the original selection. This method is useful to convert from selection to vectorial polygons.

DECLARATION

```
function GetIndexFromObj(hobj: integer): integer;
```

DESCRIPTION

Returns object index from object handle. This is the inverse of GetObjFromIndex.

DECLARATION

```
function GetMemoFormattedString(hobj: integer):
AnsiString;
```

DESCRIPTION

GetMemoFormattedString applies only to iekMEMO object and returns the text as it appears. Whenever a word wrap occurs, a #10 (line feed) is inserted. Hobj = -2 represents last inserted object.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
function GetObjFromID(oID: integer): integer;
```

DESCRIPTION

GetObjFromID finds first object with the specified ID.
IDs are stored in ObjID. The returned value is the handle of the object.

Example

```
// set pen color to clRed to the first object
// that has ObjID[]=17
var
hobj: integer;
begin
  ...
  hobj := ImageEnVect1.GetObjFromID(17);
  ImageEnVect1.ObjPenColor[hobj]:=clRed;
..
end;
```

ImageEnVect

Unit ImageEnVect

TImageEnVect

DECLARATION

```
function GetObjFromIndex(idx: integer): integer;
```

DESCRIPTION

GetObjFromIndex returns the object handle of the idx object.
It is useful to iterate over all vectorial objects.

Example

```
// changes to red pen color for each object
for i:=0 to ImageEnVect1.ObjectsCount-1 do
begin
  hobj := ImageEnVect1.GetObjFromIndex( i );
  ImageEnVect1.ObjPenColor[ hobj ] := clRed;
end;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
function GetObjFromName(const oName:
AnsiString):integer;
```

DESCRIPTION

GetObjFromName finds the first vectorial object with the specified name. Objects names are stored in ObjName. Strings are case sensitive. The returned value is the handle of the object.

Example

```
// set pen color to clRed for the first object
// that has ObjName[]='Jack'
var
hobj: integer;
begin
  ...
  hobj := ImageEnVect1.GetObjFromName('Jack');
  ImageEnVect1.ObjPenColor[hobj]:=clRed;
  ..
end;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure GetObjRect(hobj: integer; var Rect:
TRect);
```

DESCRIPTION

GetObjRect returns the coordinates of hobj object.

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

DECLARATION

```
property MemoEdit: TIETextControl;
```

DESCRIPTION

MemoEdit is undocumented.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
function MemoEditingGetCharInfo:
TIEMemoEditCharInfo;
```

DESCRIPTION

MemoEditingGetCharInfo returns the char information (font, color, alignment) about the currently caret position.

Demo

Imageprocessing1\advancedtext

DECLARATION

```
procedure MemoEditingSetCharInfo(info:
TIEMemoEditCharInfo);
```

DESCRIPTION

MemoEditingSetCharInfo sets the char information (font, color, alignment) about the currently caret position.

Demo

Imageprocessing1\advancedtext

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure ObjSetTBitmap(hobj: integer; bmp:
TBitmap);
```

DESCRIPTION

ObjSetTBitmap assigns (copies) a TBitmap object to the specified image object.

DECLARATION

```
property ObjTextEditMode: integer
```

DESCRIPTION

This property switches the specified MEMO or TEXT object in edit mode. You can use it to know which text object is currently editing.

DECLARATION

```
procedure RemoveAllObjects;
```

DESCRIPTION

RemoveAllObjects removes all objects.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure RemoveObject(hobj: integer);
```

DESCRIPTION

RemoveObject removes the hobj object. Hobj = -2 represents last inserted object.

DECLARATION

```
function RemovePolygonJaggedEdges(hobj: integer): Boolean;
```

DESCRIPTION

RemovePolygonJaggedEdges eliminates the jagged edges from a polygon created using CreatePolygonFromEdge method. This function can smooth angles.

RemovePolygonJaggedEdges returns false if fails.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure RotateAllObjects(angle: double; center: TIERotateCenter);
```

DESCRIPTION

RotateAllObjects rotates all objects by the specified angle (in degrees).

If center is ierImage only 90/180/270 degrees rotations are allowed. iekBOX, iekELLIPSE, iekBITMAP, iekTEXT can be rotated only by 90/180/270 degrees. This method doesn't work with iekMEMO and curved texts. It is better to call RotateAllObjects before Proc.Rotate.

Example

```
// rotate the image and objects by 90 degrees
ImageEnVect1.RotateAllObjects(90, ierImage);
ImageEnVect1.Proc.Rotate(90);
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure RotateObject(hobj: integer; angle:
double; center: TIERotateCenter);
```

DESCRIPTION

RotateObject rotates specified object by the specified angle (in degrees).

If center is ierImage only 90/180/270 degrees rotations are allowed. iekBOX, iekELLIPSE, iekBITMAP, iekTEXT can be rotated only by 90/180/270 degrees. This method doesn't work with iekMEMO and curved texts. Hobj = -2 represents last inserted object.

Example

```
// rotate the image and object 'hobj' by 90
// degrees
ImageEnVect1.Proc.Rotate(90);
ImageEnVect1.RotateObject(hobj, 90, ierImage);
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure SetObjBackTo(hobj: integer; refobj:
integer);
```

DESCRIPTION

SetObjBackTo visually moves object hobj behind refobj. That is, refobj will appear to be in front of hobj. If refobj is -1, SetObjBackTo visually moves hobj behind all objects.

Example

```
...
obj1 := AddNewObject;
...
obj2 := AddNewObject; // obj2 is over obj1
...
ImageEnVect1.SetObjFrontOf( obj1,obj2 );
// obj1 is over obj2

ImageEnVect1.SetObjBackTo( obj1,obj2 );
// now obj2 is over obj1

// Moves obj1 in front of all objects
ImageEnVect1.SetObjFrontOf(obj1,-1);

// Moves obj1 back to all objects
ImageEnVect1.SetObjBackTo(obj1,-1);
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure SetObjBitmapICO(hobj: integer; ico:
integer; iwidth, iheight: integer);
```

DESCRIPTION

SetObjBitmapICO sets the hobj (iekBITMAP) image as standard windows icon. ico can be OIC_SAMPLE, OIC_HAND, OIC_QUES, OIC_BANG, OIC_NOTE, OIC_WINLOGO, OIC_WARNING, OIC_ERROR, OIC_INFORMATION (look at windows.pas). Iwidth and iheight are icon width and height.

Example

```
// Sets system icon OIC_HAND of 64x64 as images
// of next inserted object iekBITMAP
ImageEnVect1.SetObjBitmapICO(-1,OIC_HAND,64,64);
```

ImageEnVect

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure SetObjFont(hobj: integer; v: TFont);
```

DESCRIPTION

SetObjFont sets the font of hobj (iekTEXT) object.
Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure SetObjFrontOf(hobj: integer; refobj:
integer);
```

DESCRIPTION

SetObjFrontOf visually moves object hobj in front of refobj.
If refobj is -1, SetObjFrontOf moves hobj in front of all objects.

Example

```
...
obj1 := AddNewObject;
....
obj2 := AddNewObject; // obj2 is over obj1
...
ImageEnVect1.SetObjFrontOf( obj1,obj2 );
// obj1 is over obj2

ImageEnVect1.SetObjBackTo( obj1,obj2 );
// now obj2 is over obj1

// Moves obj1 in front of all objects
ImageEnVect1.SetObjFrontOf(obj1,-1);

// Moves obj1 back to all objects
ImageEnVect1.SetObjBackTo(obj1,-1);
```

ImageEnVect

DECLARATION

```
procedure SetObjPolylinePoints(hobj: integer;
Points: array of TPoint);
```

DESCRIPTION

SetObjPolylinePoints specifies the points that compose the polyline for the polyline object hobj. The points are specified in bitmap coordinates.

DECLARATION

```
procedure SetObjRect(hobj: integer; const Rect:
TRect);
```

DESCRIPTION

SetObjRect specifies the coordinates of hobj object. Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

DECLARATION

```
procedure SetObjTextCurve(hobj: integer; x,y:
double);
```

DESCRIPTION

SetObjTextCurve inserts a new point into the specified curved text (iekTEXT object). Coordinates must be floating point values from 0 to 1.

Example

```
ImageEnVect.SetObjTextCurveShape(hobj, iecNone,
0,false); // remove old curve
ImageEnVect.SetObjTextCurve(hobj, 0, 0.1);
ImageEnVect.SetObjTextCurve(hobj, 0.1, 0.2);
ImageEnVect.SetObjTextCurve(hobj, 0.3, 0.4);
Etc..
```

DECLARATION

```
procedure SetObjTextCurveFromPolyline(hobj:
integer; source: integer);
```

DESCRIPTION

SetObjTextCurveFromPolyline creates a curve (calling SetObjTextCurve) with the shape specified by the source polyline object which must be a iekPOLYLINE object.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure SetObjTextCurveShape(hobj: integer;
shape: TIECurve; phase: integer; dforward:
Boolean);
```

DESCRIPTION

SetObjTextCurveShape creates a curve (calling SetObjTextCurve) with the shape specified by the shape parameter. Phase specifies the angle where the text starts on the curve (0-360), Dforward specifies the direction of the text on the curve.

Example

```
ImageEnVect.SetObjTextCurveShape( hobj,
iecEllipse, 90, true );
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure SimplifyPolygon(hobj: integer;
MaxPoints: integer);
```

DESCRIPTION

SimplifyPolygon approximates a high resolution polyline with a smaller low resolution polyline having fewer vertices (MaxPoints).

Example

```
hobj :=
ImageEnVect1.CreatePolygonFromEdge(100,100,false,25);
// Creates a polygon along the edges of the image
// at 100,100 position
ImageEnVect1.SimplifyPolygon(.hobj, 30);
// simplify the polygon to max 30 points
```

Rendering and Copying

DECLARATION

```
procedure CopyAllObjectsTo(Dest: TImageEnVect);
```

DESCRIPTION

Call CopyAllObjectsTo to write all objects to Dest TImageEnVect component.

DECLARATION

```
procedure CopyObjectsToBack(Antialias:
Boolean=true);
```

DESCRIPTION

CopyObjectsToBack copies all object over background image. This method is a way to convert the vectorial objects to a pixmap image.
If Antialias is true an anti-alias filter is applied to remove pixels aliasing.

Example

```
// Saves background image and vectorial objects
// in a BMP file
ImageEnVect1.CopyObjectsToBack(true);
ImageEnVect1.RemoveAllObjects;
ImageEnVect1.IO.SaveToFile('output.bmp');
```

Unit ImageEnVect

TImageEnVect

Declaration

```
procedure CopyObjectToBack(hobj: integer;
Antialias: Boolean = true);
```

Description

This method copies/merges the specified object over the background image. Antialias is true if you want to apply the antialias filter. Hobj = -2 represents last inserted object.

Declaration

```
function CopyObjectTo(hobj: integer; Dest:
TImageEnVect):integer;
```

Description

Call CopyObjectTo to write only the hobj object to the Dest TImageEnVect component. CopyObjectTo returns the handle of the created object.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure DrawObjectsToBitmap(target: TIEBitmap;
Antialias: Boolean = true);
```

DESCRIPTION

DrawObjectsToBitmap draws all vectorial objects on the specified TIEBitmap object (target). Antialias parameter controls the anti-alias filter.

Example

```
ImageEnVect.DrawObjectsToBitmap(
ImageEnView2.IEBitmap, true );
ImageEnView2.Update;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure DrawOneObjectToBitmap(hobj: integer;
target: TIEBitmap; Antialias: Boolean=true);
```

DESCRIPTION

DrawOneObjectToBitmap draws a vectorial object on the specified TIEBitmap object (target). Antialias parameter controls the anti-alias filter. Hobj = -2 represents last inserted object.

Example

```
ImageEnVect.DrawObjectsToBitmap(
ImageEnView2.IEBitmap, true );
ImageEnView2.Update;
```

Input/output

There are four options to save an image with its objects when using TImageEnVect:

```
1)  If you want to save only objects:
ImageEnVect1.SaveToFileIEV('file.iev');
To load:
ImageEnVect1.LoadfromFileIEV('file.iev');

2) If you want save objects and image:
ImageEnVect1.SaveToFileAll('file.all');
To load:
ImageEnVect1.LoadFromFileAll('file.all');

3) If you want save objects and image as standard
tiff and using Imaging Annotations (readable by
Windows Preview):
ImageEnVect1.IO.SaveToFile('file.tif');
ImageEnVect1.SaveObjectsToTIFF('file.tif');

To load:
ImageEnVect1.IO.LoadFromFile('file.tif');
ImageEnVect1.LoadObjectsFromTIFF('file.tif');

4) If you want save objects inside a jpeg or
other formats which do not support Imaging
Annotations or you just want to merge image and
objects:
  ImageEnVect1.CopyObjectsToBack;
  ImageEnVect1.IO.SaveToFile('file.jpg');
```

Unit ImageEnVect
TImageEnVect

DECLARATION

```
procedure ImportDXF(const FileName: WideString);
```

DESCRIPTION

ImportDXF imports an Autocad DXF file. Returns true on successful import. Note: only a subset of the DXF is implemented (lines, arcs and ellipses).

Example

```
ImageEnVect1.ImportDXF('cad.dxf');
```

DECLARATION

```
function LoadFromFileAll(const fileName:
string):Boolean;
```

DESCRIPTION

This method loads all layers and vectorial objects saved using SaveToFileAll. You cannot use this method to load standard image files (jpeg, tiff...).

Unit ImageEnVect

TImageEnVect

DECLARATION

```
function LoadFromFileIEV(const FileName: string): Boolean;
```

DESCRIPTION

LoadFromFileIEV loads the objects from the specified file. The IEV format is used to save vectorial objects as lines, ellipses and bitmaps (but not background image). LoadFromFileIEV returns true if successful.

Example

```
ImageEnVect1.LoadFromFileIEV('objects.iev');
```

DECLARATION

```
function LoadFromStreamAll(Stream: TStream): Boolean;
```

DESCRIPTION

This method loads all layers and vectorial objects saved using SaveToStreamAll.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure LoadFromStreamIEV(Stream: TStream);
```

DESCRIPTION

LoadFromStreamIEV loads objects from a stream. The IEV format is written to save vectorial objects as lines, ellipses and bitmaps (but not background image). LoadFromStreamIEV returns true on successful.

Example

```
// Loads two vectorial images from "maps.iev",
// and shows them in ImageEnVect1 and
// ImageEnVect2
var
fs: TFileStream;
begin
  fs :=TFileStream.Create('maps.iev',fmOpenRead);
  ImageEnVect1.LoadFromStreamIEV(fs);
  ImageEnVect2.LoadFromStreamIEV(fs);
  fs.free;
end;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure LoadObjectsFromTIFF(const fileName:
string; pageIndex: integer);
```

DESCRIPTION

This method loads objects from the specified TIFF. This is like LoadFromFileIEV, but gets the objects info from a tag of the TIFF file. This method is not compatible with Wang Imaging (you have to use another method to save as Wang Imaging), but allows you to save all TImageEnVect objects. PageIndex specifies the page in a multi-page tiff used to store the objects. In this way you can store a set of objects for each tiff page.

The tag used for default is 40101. However you can change it rewriting the value in iegObjectsTIFFTag public variable (defined in ImageEnIO unit). Example: iegObjectsTIFFTag:=49001. To save objects use SaveObjectsToTIFF.

Example

```
// saves the background image
ImageEnVect1.IO.SaveToFile('output.tif');
// saves the objects
ImageEnVect1.SaveObjectsToTIFF('output.tif',0);

// loads the background image
ImageEnVect1.IO.LoadFromFile('output.tif');
// loads the objects
ImageEnVect1.LoadObjectsFromTIFF('output.tif',0);
```

Unit ImageEnVect
TImageEnVect

DECLARATION

```
procedure SaveObjectsToTIFF(const fileName:
string; pageIndex: integer);
```

DESCRIPTION

This method saves all objects in the specified TIFF. This is like SaveToFileIEV, but incorporates the objects info in a tag of the TIFF file. This method is not compatible with Wang Imaging (you have to use another method to save as Wang Imaging), but allows you to save all TImageEnVect objects. PageIndex specifies the page in a multi-page tiff used to store the objects. In this way you can store a set of objects for each tiff page.

The tag used for default is 40101. However you can change it rewriting the value in iegObjectsTIFFTag public variable (defined in ImageEnIO unit). Example: iegObjectsTIFFTag:=49001;

To load objects back use LoadObjectsFromTIFF.

Example

```
// saves the background image
ImageEnVect1.IO.SaveToFile('output.tif');
// saves the objects
ImageEnVect1.SaveObjectsToTIFF('output.tif',0);

// loads the background image
ImageEnVect1.IO.LoadFromFile('output.tif');
// loads the objects
ImageEnVect1.LoadObjectsFromTIFF('output.tif',0);
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure SaveToFileAll(const fileName: string;
imageCompression: TIOFileType);
```

DESCRIPTION

This method saves all layers and vectorial objects in one single file. This is like a consecutive LayersSaveToXXX and SaveToFileIEV. ImageCompression specifies how compress the background image and the layers (cannot be ioTIFF). If ImageCompression is -1, the image is saved using an internal format which preserves pixel format and alphachannel.

Example

```
ImageEnVect1.SaveToFileAll('file.my', );
..
ImageEnVect1.LoadFromFileAll('file.my');
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure SaveToFileIEV(const FileName: string);
```

DESCRIPTION

SaveToFileIEV saves objects to the specified file. The IEV format is written to save vectorial objects as lines, ellipses and bitmaps (but not background image).

Example

```
ImageEnVect1.SaveToFileIEV('objects.iev');
```

DECLARATION

```
procedure SaveToStreamAll(Stream: TStream;
imageCompression: TIOFileType);
```

DESCRIPTION

This method saves all layers and vectorial objects in one single stream block. This is like a consecutive LayersSaveToXXX and SaveToFileIEV.

ImageCompression specifies how compress the background image and the layers (cannot be ioTIFF). If ImageCompression is -1, the image is saved using an internal format which preserves pixel format and alphachannel.

Unit ImageEnVect

TImageEnVect

DESCRIPTION

```
procedure SaveToStreamIEV(Stream: TStream);
```

DESCRIPTION

SaveToStreamIEV saves objects to file. The IEV format is written to save vectorial objects as lines, ellipses and bitmaps (but not background image).

Example

```
// saves the vectorial images contained in
// ImageEnVect1 and ImageEnVect2 to "mappe.dat".
var
fs: TFileStream;
begin
  fs := TFileStream.Create('mappe.dat',fmCreate);
  ImageEnVect1.SaveToStreamIEV(fs);
  ImageEnVect2.SaveToStreamIEV(fs);
  fs.free;
end;
```

Unit ImageEnVect

TImageEnVect

DESCRIPTION

```
procedure SelAllObjects;
```

DESCRIPTION

SelAllObjects select all objects.

DECLARATION

```
function SetObjBitmapFromFile(hobj: integer;
const FileName: WideString): Boolean;
```

DESCRIPTION

SetObjBitmapFromFile loads an image from FileName and assigns it to the hobj bitmap object.

It returns true if a file is successfully loaded. This function also loads and assigns the alpha channel if present.

Hobj = -2 represents last inserted object.

Example

```
ImageEnVect1.SetObjBitmapFromFile(-1, 'image.gif' );
ImageEnVect1.ObjKind[-1]:=iekBitmap;
ImageEnVect1.AddNewObject;

// Load the alpha channel from PNG (and others)
ImageEnVect1.SetObjBitmapFromFile(hobj, 'test.png');
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
function SetObjBitmapFromStream(hobj: integer;
Stream: TStream; FileFormat: TIOFileType):
Boolean;
```

DESCRIPTION

SetObjBitmapFromStream loads an image from Stream and assigns it to the hobj bitmap object.

It returns true if a file is successfully loaded. This function also loads and assigns the alpha channel if present. If you set FileFormat = ioUnknown then the format is autodetected.
Look also SetObjBitmapFromFile.

Clipboard

Declaration

```
procedure ObjCopyToClipboard;
```

Description

ObjCopyToClipboard copies selected objects to the clipboard in a proprietary format.

See also
ObjCutToClipboard
ObjIsClipboardAvailable
ObjPasteFromClipboard

Declaration

```
procedure ObjCutToClipboard;
```

Description

An ObjCutToClipboard copy selected objects to the clipboard in a proprietary format and then deletes the objects.

See also
ObjCopyToClipboard
ObjIsClipboardAvailable
ObjPasteFromClipboard

Unit ImageEnVect

TImageEnVect

DECLARATION

```
function ObjIsClipboardAvailable: Boolean;
```

DESCRIPTION

This method returns true if clipboard contains data valid for TImageEnVect.

See also

ObjCopyToClipboard
ObjCutToClipboard
ObjPasteFromClipboard

DECLARATION

```
procedure ObjPasteFromClipboard(OffsetX, OffsetY: integer);
```

DESCRIPTION

ObjPasteFromClipboard copies the contents from the clipboard into the TImageEnVect component, replacing the currently selected object(s).

Parameter	Description
OffsetX	Vertical offset from original object's position.
OffsetY	Horizontal offset from original object's position.

See also
ObjCopyToClipboard
ObjCutToClipboard
ObjIsClipboardAvailable

Object Properties

DECLARATION

```
property ObjAngleShowSmall: Boolean;
```

DESCRIPTION

If True angle objects shows only the small angle.

DECLARATION

```
property ObjArcEndingAngle[hobj: integer]:
double;
```

DESCRIPTION

ObjArcEndingAngle specifies the ending angle in radians for an iekARC object.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjArcStartingAngle[hobj: integer]: double;
```

DESCRIPTION

ObjArcStartingAngle specifies the starting angle in radians for an iekARC object.

DECLARATION

```
property ObjAspectRatio[hobj: integer]: Boolean;
```

DESCRIPTION

When true, the object maintains the aspect ratio when resized (like ALT key).

DECLARATION

```
property ObjBeginShape[hobj: integer]: TIEShape;
```

DESCRIPTION

Use ObjBeginShape to specify the beginning shape of hobj object (iekLINE object type).

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Example

```
// next object (iekLINE) has two out-arrows to
// the extremities
ImageEnVect1.ObjBeginShape[-1]:=iesOUTARROW;
ImageEnVect1.ObjEndShape[-1]:=iesOUTARROW;
```

DECLARATION

```
property ObjBitmapAlpha[hobj: integer]:
TIEBitmap;
```

DESCRIPTION

ObjBitmapAlpha contains the alpha channel of the specified object (which must be a iekBitmap object). You can obtain the same value using ObjBitmap[hobj].AlphaChannel.

DECLARATION

```
property ObjBitmapBorder[hobj: integer]: Boolean;
```

DESCRIPTION

If ObjBitmapBorder is true a border around the bitmap object will be painted.

DECLARATION

```
property ObjBitmap[hobj: integer]: TIEBitmap;
```

DESCRIPTION

ObjBitmap is the image (bitmap) that hobj shows. Hobj is iekBITMAP type. If ShareBitmaps is true, all identical images are stored in the some memory space (this frees much memory). However ObjBitmap always makes a copy of the specified TIEBitmap object. Hobj is a value identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Example

```
// Copy the image of ImageEnView1 to the next
// object to insert
ImageEnVect1.ObjBitmap[-
1]:=ImageEnView1.IEBitmap;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjBlendOperation[hobj: integer]:
TIERenderOperation;
```

DESCRIPTION

ObjBlendOperation specifies the blending operation. Default is ielNormal. Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

DECLARATION

```
property ObjBoxHighlight[hobj: integer]: Boolean;
```

DESCRIPTION

ObjBoxHighlight was introduced to increase support for Imaging Annotations. When ObjBoxHighlight is true, it makes a filled box as a highlight box (highlight the background with the fill color).

Hobj = -2 represents last inserted object.

DECLARATION

```
property ObjBoxInnerSelectable: Boolean;
```

DESCRIPTION

When false (default) iekBOX objects are selectable only clicking on the border, unless ObjBrushStyle is bsSolid. When true iekBOX objects are always selectable inside the bounding box (even with ObjBrushStyle = bsClear).

DECLARATION

```
property ObjBrushColor[hobj: integer]:  TColor;
```

DESCRIPTION

ObjBrushColor is the brush color of hobj object. hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Example

```
// Set clRed as brush (background) color for the
// next object to insert
ImageEnVect1.ObjBrushColor[-1]:=clRed;
```

Unit ImageEnVect

TImageEnVect

Declaration

```
property ObjBrushStyle[hobj: integer]:
TBrushStyle;
```

Description

ObjBrushStyle is the brush style of hobj object. Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Example

```
// Sets bsSolid as brush style for the next
// object to insert
ImageEnVect1.ObjBrushStyle[-1]:=bsSolid;
```

DECLARATION

```
property ObjEndShape[hobj: integer]: TIEShape;
```

DESCRIPTION

ObjEndShape is the end shape of hobj object (iekLINE).
Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Example

```
// These set the next iekLINE object sides to
// out-arrow
ImageEnVect1.ObjBeginShape[-1]:=iesOUTARROW;
ImageEnVect1.ObjEndShape[-1]:=iesOUTARROW;
```

DECLARATION

```
property ObjFontAngle[hobj: integer]: double;
```

DESCRIPTION

ObjFontAngle is the rotation angle of iekTEXT object hobj. The angle is in degrees (positive values rotate counter clockwise).

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

DECLARATION

```
property ObjFontHeight[hobj: integer]: integer;
```

DESCRIPTION

ObjFontHeight specifies the font height for iekTEXT object hobj. Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Run-time only

DECLARATION

```
property ObjFontLocked[hobj: integer]: Boolean;
```

DESCRIPTION

When the object is a Memo (iekMemo), ObjFontLocked locks the alignment, color and font, allowing all characters to have the same visual characteristics.

If the ObjFontLocked property is disabled (False) the user can modify font and alignment using the control keys in iegMemoShortCuts.

DECLARATION

```
property ObjFontName[hobj: integer]: AnsiString;
```

DESCRIPTION

ObjFontName is the font name for iekTEXT object hobj. Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Example

```
// Sets 'Arial' as font type for next object to
// insert
ImageEnVect1.ObjFontName[-2]:='Arial';
```

DECLARATION

```
property ObjFontStyles[hobj: integer]:
TFontStyles;
```

DESCRIPTION

ObjFontStyles sets the font style for the iekTEXT object hobj. Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Example

```
ImageEnVect1.ObjFontStyles[-1]:=fsBold;
```

Unit ImageEnVect

TImageEnVect

Declaration

```
property ObjFontQuality[hobj: integer]:
TIEFontQuality;
```

Description

ObjFontQuality specifies the quality (antialias, clear type, etc.) of the text font.

Declaration

```
property ObjHeight[hobj: integer]: integer;
```

Description

ObjHeight specifies the height of hobj object. The coordinates are in pixels with zoom = 100%. Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Example

```
// This code creates a Box at 10, 10 of 50x50
// pixels.
ImageEnVect1.ObjKind[-1]:=iekBOX;
ImageEnVect1.ObjLeft[-1]:=10;
ImageEnVect1.ObjTop[-1]:=10;
ImageEnVect1.ObjWidth[-1]:=50;
ImageEnVect1.ObjHeight[-1]:=50;
ImageEnVect1.AddNewObject;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjID[hobj: integer]: integer;
```

DESCRIPTION

ObjID is a identification value for the object hobj. ImageEn doesn't use this value directly, but saves and loads it like other properties.

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

DECLARATION

```
function ObjIsVisible(hobj: integer): Boolean;
```

DESCRIPTION

Returns true if the specified object is currently visible.

DECLARATION

```
property ObjKind[hobj: integer]: TIEVObjectKind;
```

DESCRIPTION

ObjKind is the type (line, ellipse...) of object hobj.
Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Declaration

```
property ObjLabelBorder[hobj: integer]:
TIELabelBorder;
```

Description

ObjLabelBorder is the label (text) border of hobj object (iekLINELABEL).

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Declaration

```
property ObjLabelBrushColor[hobj: integer]:
TColor;
```

Description

ObjLabelBrushColor is the brush color of hobj object (iekLINELABEL). Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

DECLARATION

```
property ObjLabelBrushStyle[hobj: integer]:
TBrushStyle;
```

DESCRIPTION

ObjLabelBrushStyle is the brush style of hobj object (iekLINELABEL). Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

DECLARATION

```
property ObjLabelPosition[hobj: integer]:
TIELabelPos;
```

DESCRIPTION

ObjLabelPosition is the label (text) position of hobj object (iekLINELABEL). Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjLayer[hobj: integer]: integer;
```

DESCRIPTION

ObjLayer specifies the layer index where the object is located (drawn and referenced). The default is "0". When ObjLayer is 0 ImageEnVect draws the object on layer 0. When ObjLayer is 1 ImageEnVect draws the object on layer 1.. and so on.

Hobj is the ID of the object. You can specify hobj as IEV_NEXT_INSERTED_OBJECT (-1) which refers to the next object to be inserted or IEV_PREVIOUS_INSERTED_OBJECT (-2) for the last object inserted.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjLeft[hobj: integer]: integer;
```

DESCRIPTION

ObjLeft specifies the left offset of hobj object. The coordinates are in pixels with zoom = 100%. Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Example

```
// This code creates a Box at 10, 10 of 50x50
// pixels.
ImageEnVect1.ObjKind[-1]:=iekBOX;
ImageEnVect1.ObjLeft[-1]:=10;
ImageEnVect1.ObjTop[-1]:=10;


ImageEnVect1.ObjWidth[-1]:=50;
ImageEnVect1.ObjHeight[-1]:=50;
ImageEnVect1.AddNewObject;
```

Declaration

```
property ObjMemoBorderColor[hobj: integer]:
TColor;
```

Description

ObjMemoBorderColor specifies the color of the memo border.

Declaration

```
property ObjMemoBorderStyle[hobj: integer]:
TPenStyle;
```

Description

ObjMemoBorderStyle specifies the style of the memo border.

DECLARATION

```
property ObjMemoCharsBrushStyle[hobj: integer]:
TBrushStyle;
```

DESCRIPTION

This property allows setting the default brush style when ObjFontLocked is false. The default is bsSolid, which means that when you change the memos background color the characters remain with their unchanged background color.

DECLARATION

```
property ObjMemoFixedHeight[hobj: integer]:
integer;
```

DESCRIPTION

ObjMemoFixedHeight specifies the interline space between lines. By setting a value of 0 (default), ObjMemoFixedHeight is automatically calculated.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjMemoHasBitmap[hobj: integer]:
Boolean;
```

DESCRIPTION

When true, the memo object has a background image. You can set the bitmap using ObjBitmap, ObjSetTBitmap, SetObjBitmapICO and SetObjBitmapFromFile methods. The ObjBrushStyle must be bsClear and fonts must not have filled background.

DECLARATION

```
property ObjMemoLineSpace[hobj: integer]:
integer;
```

DESCRIPTION

ObjMemoLineSpace specifies the interline space. 0 = automatically calculated.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjMemoMarginBottom[hobj: integer]:
double;
```

DESCRIPTION

ObjMemoMarginBottom specifies the bottom margin in percentage of vertical height. This applies only to memo objects.

Example

```
ImageEnVect1.ObjMemoMarginLeft[h]:=10;
// 10% left
ImageEnVect1.ObjMemoMarginTop[h]:=10;
// 10% top
ImageEnVect1.ObjMemoMarginRight[h]:=10;
// 10% right
ImageEnVect1.ObjMemoMarginBottom[h]:=10;
// 10% bottom
```

Unit ImageEnVect

TImageEnVect

Declaration

```
property ObjMemoMarginLeft[hobj: integer]:
double;
```

Description

ObjMemoMarginLeft specifies the left margin in percentage of horizontal width. This applies only to memo objects.

Example

```
ImageEnVect1.ObjMemoMarginLeft[h]:=10;
// 10% left
ImageEnVect1.ObjMemoMarginTop[h]:=10;
// 10% top
ImageEnVect1.ObjMemoMarginRight[h]:=10;
// 10% right
ImageEnVect1.ObjMemoMarginBottom[h]:=10;
// 10% bottom
```

DECLARATION

```
property ObjMemoMarginRight[hobj: integer]:
double;
```

DESCRIPTION

ObjMemoMarginRight specifies the right margin in percentage of horizontal width. This applies only to memo objects.

Example

```
ImageEnVect1.ObjMemoMarginLeft[h]:=10;
// 10% left
ImageEnVect1.ObjMemoMarginTop[h]:=10;
// 10% top
ImageEnVect1.ObjMemoMarginRight[h]:=10;
// 10% right
ImageEnVect1.ObjMemoMarginBottom[h]:=10;
// 10% bottom
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjMemoMarginTop[hobj: integer]: double;
```

DESCRIPTION

ObjMemoMarginTop specifies the top margin in percentage of vertical height. This applies only to memo objects.

Example

```
ImageEnVect1.ObjMemoMarginLeft[h]:=10;
// 10% left
ImageEnVect1.ObjMemoMarginTop[h]:=10;
// 10% top
ImageEnVect1.ObjMemoMarginRight[h]:=10;
// 10% right
ImageEnVect1.ObjMemoMarginBottom[h]:=10;
// 10% bottom
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjName[hobj: integer]: AnsiString
```

DESCRIPTION

ObjName is an application string for the object hobj. ImageEn doesn't use this value, but saves and loads it like other properties.

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

DECLARATION

```
property ObjPenColor[hobj: integer]: TColor;
```

DESCRIPTION

ObjPenColor specifies the pen color for hobj object.

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjPenStyle[hobj: integer]: TPenStyle;
```

DESCRIPTION

ObjPenStyle sets the pen style of hobj object.

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

DECLARATION

```
property ObjPenWidth[hobj: integer]: integer;
```

DESCRIPTION

ObjPenWidth sets the pen width of hobj object.

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjPolylineClosed[hobj: integer]:
Boolean;
```

DESCRIPTION

ObjPolylineClosed is true if we want close the polyline (so it is like a polygon).

DECLARATION

```
property ObjPolylinePointsCount[hobj: integer]:
integer;
```

DESCRIPTION

ObjPolylinePointsCount specifies the count of points that compose the hobj polyline. Use ObjPolylinePoints to obtain the coordinates of point that compose the polyline.

DECLARATION

```
property ObjPolylinePoints[hobj: integer; index: Integer]: TPoint;
```

DESCRIPTION

ObjPolylinePoints specifies the points that compose the hobj polyline. Use ObjPolylinePointsCount to know how many points the polyline contains.

DECLARATION

```
property ObjRulerQuoteHorizon: Boolean;
```

DESCRIPTION

If True (default) ruler-quote objects maintain text on horizon.

DECLARATION

```
property ObjRulerType[hobj: integer]: TIEVRulerType;
```

DESCRIPTION

ObjRulerType specifies the type of the ruler object hobj.

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

DECLARATION

```
property ObjRulerUnit[hobj: integer]: TIEUnits;
```

DESCRIPTION

ObjRulerUnit specifies the measurement unit of the ruler object hobj.

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

DECLARATION

```
property ObjShapeHeight[hobj: integer]: integer;
```

DESCRIPTION

ObjShapeHeight sets the shape height of iekLINE hobj object. The shape is set with ObjBeginShape and ObjEndShape properties.

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

See also
ObjShapeWidth.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjShapeWidth[hobj: integer]: integer;
```

DESCRIPTION

ObjShapeWidth sets the shape width of iekLINE hobj object. The shape is set with ObjBeginShape and ObjEndShape properties.

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

See also

ObjShapeHeight

DECLARATION

```
property ObjSoftShadow[hobj: integer]:
TIEVSoftShadow;
```

DESCRIPTION

ObjSoftShadow enables and sets the object shadow properties.

Example

```
ImageEnVect.ObjSoftShadow[-1].Enabled:=True;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjStyle[hobj: integer]: TIEVStyle;
```

DESCRIPTION

ObjStyle sets some properties of the hobj object.

The default value is: [ievsSelectable, ievsMoveable, ievsSizeable, ievsVisible]. Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Example

```
// Hides the object called "cloud"
hobj := ImageEnVect1.GetObjFromName('cloud');
// hobj is an integer
ImageEnVect1.ObjStyle[hobj]:=ImageEnVect1.ObjStyl
e[hobj] - [ievsVisible];
```

Unit ImageEnVect

TImageEnVect

Declaration

```
property ObjTextAlign[hobj: integer]:
TIEAlignment;
```

Description

ObjTextAlign sets the text alignment of hobj object (iekTEXT).

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted). IejJustify applies only to MEMO objects.

Example

```
// Centers the text of next object to insert
ImageEnVect1.ObjTextAlign[-1] := iejLeft;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjTextAutoSize[hobj: integer]: Boolean;
```

DESCRIPTION

Use ObjTextAutoSize to make the iekText object adjust its size automatically so the bounding box accommodates the width of the text.

When ObjTextAutoSize is false, the text object has a fixed width. When ObjTextAutoSize is true, the size of the object is readjusted whenever the user inserts or deletes characters.

Hobj = -2 represents last inserted object.

DECLARATION

```
property ObjTextCurveCharRot[hobj: Integer]:
double;
```

DESCRIPTION

ObjTextCurveCharRot specifies the angle of each character for curved text. By specifying a value of -1, the angle is auto-calculated (follows the curve).

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjTextEditable[hobj: integer]: Boolean;
```

DESCRIPTION

If true (default) the specified text or memo object is editable, otherwise it is read only.

DECLARATION

```
property ObjText[hobj: integer]: AnsiString;
```

DESCRIPTION

ObjText is the text shown by hobj object (iekTEXT).

Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjTop[hobj: integer]: integer;
```

DESCRIPTION

ObjTop specifies the top offset of hobj object. The coordinates are in pixels with zoom = 100%. Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Example

```
// This code creates a Box at 10, 10 of 50x50
pixels.
ImageEnVect1.ObjKind[-1]:=iekBOX;
ImageEnVect1.ObjLeft[-1]:=10;
ImageEnVect1.ObjTop[-1]:=10;
ImageEnVect1.ObjWidth[-1]:=50;
ImageEnVect1.ObjHeight[-1]:=50;
ImageEnVect1.AddNewObject;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjTransparency[hobj: integer]: integer;
```

DESCRIPTION

ObjTransparency sets the transparency of the specified object. 0=fully transparent, 255=fully opaque.

DECLARATION

```
property ObjUserDataLength[hobj: integer]: integer;
```

DESCRIPTION

Applications can store custom data ObjUserData. ObjUserData contains a pointer to a user buffer which length is in ObjUserDataLength. Load/save and clipboard methods save this field allocating the buffer when needed and freeing it when an object is destroyed.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjUserData[hobj: integer]: pointer;
```

DESCRIPTION

Applications can store custom data ObjUserData. ObjUserData contains a pointer to a user buffer which length is in ObjUserDataLength. Load/save and clipboard methods save this field allocating the buffer when needed and freeing it when an object is destroyed.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjWidth[hobj: integer]: integer;
```

DESCRIPTION

ObjWidth specifies the width of hobj object. The coordinates are in pixels with zoom = 100%. Hobj is a value that identifies the object. Hobj can be -1 (next object to insert) or -2 (last object inserted).

Example

```
// This code creates a Box at 10, 10 of 50x50
pixels.
ImageEnVect1.ObjKind[-1]:=iekBOX;
ImageEnVect1.ObjLeft[-1]:=10;
ImageEnVect1.ObjTop[-1]:=10;
ImageEnVect1.ObjWidth[-1]:=50;
ImageEnVect1.ObjHeight[-1]:=50;
ImageEnVect1.AddNewObject;
```

Grips

DECLARATION

```
property ObjGripBrush: TBrush;
```

DESCRIPTION

Specifies the grip's brush as shown when an object is selected.

DECLARATION

```
property ObjGripImage: TPicture;
```

DESCRIPTION

ObjGripImage allows setting a custom picture for grips (objects painted around an object to handle resizing and selection).

DECLARATION

```
property ObjGripPen: TPen;
```

DESCRIPTION

Specifies the grip's pen as shown when an object is selected.

DECLARATION

```
property ObjGripShape: TIEGripShape;
```

DESCRIPTION

Specifies the grip's shape as shown when an object is selected.

DECLARATION

```
property ObjGripSize: integer;
```

DESCRIPTION

Specifies the grip's size as shown when an object is selected.

Selection

DECLARATION

```
procedure AddSelObject(hobj: integer);
```

DESCRIPTION

AddSelObject insert hobj object into the selected objects list.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
procedure CopySelectedObjectsTo(Dest:
TImageEnVect);
```

DESCRIPTION

Call CopySelectedObjectsTo to write all selected objects to Dest TImageEnVect component.

DECLARATION

```
function IsSelObject(hobj: integer): Boolean;
```

DESCRIPTION

IsSelObject returns true if hobj object is selected.
Last inserted object (-1) is always selected.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property MaxSelectionDistance: integer;
```

DESCRIPTION

MaxSelectionDistance indicates the maximum distance (in bitmap pixels) from which an object may be selected. If the user left clicks at a location beyond this distance, no object will be selected.

A value of -1 (the default) disables the maximum selection distance calculation (a click always selects some object).

Example

```
ImageEnVect1.MaxSelectionDistance:=-1;
// disables maximum distance

ImageEnVect1.MaxSelectionDistance:=10;
// set maximum distance to 10 bitmap pixels
```

DECLARATION

```
property SelObjects[idx: integer]: integer;
```

DESCRIPTION

SelObjects returns the id (hobj) of idx th selected object. The first selected object has an idx value of zero.

Read-only

Example

```
// Sets pen color to clRed for all selected
// objects.
for i:=0 to ImageEnVect1.SelObjectsCount-1 do
  ImageEnVect1.ObjPenColor[
ImageEnVect1.SelObjects[i] ] := clGreen;
```

DECLARATION

```
property SelObjectsCount: integer;
```

DESCRIPTION

SelObjectsCount returns how many objects are selected.

Read-only

Example

```
// Sets pen color to clRed for all selected
// objects.
for i:=0 to ImageEnVect1.SelObjectsCount-1 do
  ImageEnVect1.ObjPenColor[
ImageEnVect1.SelObjects[i] ] := clGreen;
```

DECLARATION

```
procedure UnSelAllObjects;
```

DESCRIPTION

UnSelAllObjects deselects all objects.

DECLARATION

```
procedure UnSelObject(hobj: integer);
```

DESCRIPTION

UnSelObject deselects hobj object.

Undo/Redo

DECLARATION

```
property ObjAutoUndo: Boolean;
```

DESCRIPTION

If ObjAutoUndo is true, ObjSaveUndo is called automatically before the user changes objects.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjCanUndo: Boolean;
```

DESCRIPTION

ObjCanUndo is true when the Undo stack contains at least one group of objects. ObjAllClearUndo (or ObjClearUndo if there is only one group of objects) sets ObjCanUndo to false.

DECLARATION

```
procedure ObjClearAllUndo;
```

DESCRIPTION

ObjClearAllUndo empties the Undo stack.

DECLARATION

```
procedure ObjClearUndo;
```

DESCRIPTION

ObjClearUndo clears only the last undo buffer (group of saved objects).

ImageEnVect

DECLARATION

```
procedure ObjSaveUndo;
```

DESCRIPTION

ObjSaveUndo saves selected objects to the Undo stack.

DECLARATION

```
procedure ObjUndo;
```

DESCRIPTION

ObjUndo restores the most recently saved objects from the Undo stack.

DECLARATION

```
procedure ObjUndoAt(Position: integer);
```

DESCRIPTION

ObjUndoAt restores the image (group of objects) at Position in the Undo stack. Position: 0=last saved undo; 1=before last saved undo; 2... up to UndoCount-1

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ObjUndoCount: integer;
```

DESCRIPTION

ObjUndoCount returns how many images (group of objects) there are in the Undo stack.

DECLARATION

```
property ObjUndoLimit: integer;
```

DESCRIPTION

ObjUndoLimit specifies how many images (groups of objects) can be saved using ObjSaveUndo method. Default value is 1. When you call ObjSaveUndo, ImageEn pushes the current image (group of objects) onto an image stack. Calling ObjUndo causes ImageEn to restore the last saved image. Calling ObjClearUndo causes ImageEn to remove the last saved image.

Unit ImageEnVect
TImageEnVect

DECLARATION

```
property ObjUndoMode: TIEVUndoMode;
```

DESCRIPTION

This property allows sharing the Undo/Redo system between image processing and vectorial objects. The default value (ieumSeparated) separates the two systems.

Example

```
ImageEnVect1.ObjUndoMode:=ieumShared;
ImageEnVect1.ObjAutoUndo:=true;
ImageEnVect1.Proc.UndoLimit:=10;
```

From now you can do Undo of image processing or vectorial objects just calling:

```
ImageEnVect1.Proc.Undo;
ImageEnVect1.Proc.ClearUndo;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
TIEVUndoMode = (ieumSeparated, ieumShared);
```

DESCRIPTION

Value ieumShared activates the unique Undo/Redo system for image processing and vectorial objects.

ieumSeparated separates image processing and vectorial undo/redo systems.

Others

DECLARATION

```
property InsertingPen: TPen;
```

DESCRIPTION

Pen used when user inserts a line, box or ellipse object.

DECLARATION

```
property ObjectsCount: integer;
```

DESCRIPTION

Read ObjectsCount to determine the number of objects in the TImageEnVect component.

Read-Only

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property ShareBitmaps: Boolean;
```

DESCRIPTION

If ShareBitmaps is true, ShareBitmaps activates 'equal images' sharing. In this mode, each image inserted with ObjBitmap is compared (pixel per pixel) with already inserted images. If the new image is already present, it will be marked as a reference to the old image.

For these reasons each change made to a shared image will be made to all objects that use it.

Events

DECLARATION

```
property OnActivateTextEdit: TNotifyEvent;
```

DESCRIPTION

This event occurs whenever a text edit (memo or text) is activated.

Demo

Imageprocessing1\advancedtext

Declaration

```
property OnAfterDrawObject: TIEDrawObjectEvent;
```

Description

This event occurs just after an object is painted.
Handled parameter is not used. This event is useful to draw custom objects.

Declaration

```
property OnBeforeDrawObject: TIEDrawObjectEvent;
```

Description

This event occurs just before an object is painted. This event is useful to draw custom objects.

Declaration

```
property OnBeforeVectorialChanged: TNotifyEvent;
```

Description

OnBeforeVectorialChanged event occurs just before an object is added, removed or modified by a user action.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property OnDeactivateTextEdit: TNotifyEvent;
```

DESCRIPTION

This event occurs whenever a text edit (memo or text) is deactivated.

Demo
Imageprocessing1\advancedtext

DECLARATION

```
property OnDragLenEnd:TIEVDragLenEndEvent;
```

DESCRIPTION

OnDragLenEnd event is called whenever the user releases the mouse button during dynamic distance measurement task (miDragLen).

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property OnMeasureHint: TIEVMeasureHintEvent;
```

DESCRIPTION

OnMeasureHint event is called whenever the measurement hint is shown.

Example

```
// display measure value to the Label1 (and
// repaint it to perform real-time measurement).
procedure TForm1.ImageEnVect1MeasureHint(Sender:
TObject; var Text: AnsiString;
Value: Double);
begin
  Label1.Caption := Text;
  Label1.Repaint;
end;
```

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property OnNewObject: TIEVNewObject;
```

DESCRIPTION

OnNewObject is called whenever a new object is created by some user interaction.

DECLARATION

```
property OnObjectClick: TIEVObjectClickEvent;
```

DESCRIPTION

OnObjectClick is called whenever an object is clicked.

DECLARATION

```
property OnObjectDblClick: TIEVObjectClickEvent;
```

DESCRIPTION

OnObjectDblClick is called whenever an object is double clicked.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property OnObjectMoveResize:
TIEVObjectMoveResizeEvent;
```

DESCRIPTION

OnObjectMoveResize is called whenever an object is moved or resized.

DECLARATION

```
property OnObjectOver: TIEVObjectClickEvent;
```

DESCRIPTION

OnObjectOver is called whenever the mouse is over an object.

DECLARATION

```
property OnPresentMeasure: TIEOnPresentMeasure;
```

DESCRIPTION

OnPresentMeasure event is called whenever a measure must be converted to string.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property OnSelectObject: TNotifyEvent;
```

DESCRIPTION

The OnSelectObject event is called whenever an object is selected or deselected.

DECLARATION

```
property OnTextEditCursorMoved: TNotifyEvent;
```

DESCRIPTION

This event occurs whenever the cursor moves on text/memo editing.

Demo
Imageprocessing1\advancedtext

DECLARATION

```
property OnTextKeyDown: TKeyEvent;
```

DESCRIPTION

This event occurs whenever a key is pressed inside a TEXT or MEMO object.

Unit ImageEnVect

TImageEnVect

DECLARATION

```
property OnUserDeselectObject:
TIEUserSelectObject;
```

DESCRIPTION

This event occurs whenever user de-selects an object (not the application!) by means of a mouse action.

DECLARATION

```
property OnUserSelectObject: TIEUserSelectObject;
```

DESCRIPTION

This event occurs whenever user selects an object (not the application!) by means of a mouse action.

DECLARATION

```
property OnVectorialChanged: TNotifyEvent;
```

DESCRIPTION

OnVectorialChanged is called whenever an object is added, removed or modified by a user's action.

Chapter 6

TImageEnIO

Chapter 6. ImageEnIO

ImageEnIO is used for all graphic input and output. TImageEnIO handles input/output operations.

TImageEnView (ImageEnView1) already encapsulates TImageEnIO and TImageEnProc, so you don't actually need to put a TImageEnIO or a TImageEnProc component on the form.

TImageEnIO must be attached to a TImageEnView (or inherited objects), TIEBitmap, TImage and TBitmap objects. ImageEnIO can be accessed with ImageEnView1.IO instead of using the ImageEnIO1 component and ImageEnProc can be accessed with ImageEnView1.Proc instead of using the ImageEnProc1 component.

It also maintains specific file format parameters (Params) and uses them when loading or saving to file/stream.

Note: Users often attach a TImageEnIO component to a TImageEnMView component. This is not correct. The TImageEnMIO component must be attached only to a TImageEnMView component.

Loading Multi-frame files

The key to loading multi-frame file formats is setting the index to the frame to load as well as storing each frame for display or editing. Frames may be stored in memory with the IE class or in components or loaded at runtime in a single TImageEnView by setting the index of the frame to be loaded before loading the file.

A popular way to display multi-frame files is to create a TImageEnView on a TPageControl Tab sheet at runtime. When using ImageEn in this manner the properties and events for each TImageEnView must be set as well. Typically this is accomplished by determining the number of frames in the file then storing a description of the frame in a TListView and filling a TPageControl with child TImageEnView's with the images of each frame.

Accessing Each Frame

Each frame of a multi-frame image may be displayed with TImageEnView as follows:

```
procedure TForm1.OpenFile(iFilename: string);
var
  iFileType: TIOFileType;
  iFrameIndex: integer;
begin
  if FileExists(iFilename) then
  begin
    Screen.Cursor := crHourglass;
    try
      ImageEnView1.IO.ParamsFromFile(iFilename);
      iFileType :=
                IEExtToFileFormat(ExtractFileExt
                  (iFilename));
      // Get the frame index to display
      iFrameIndex := StrToInt(
                       EditFrameIndex1.Text)
      case iFileType of
        ioICO:
          ImageEnView1.IO.Params.ICO_ImageIndex
            := iFrameIndex;
        ioGIF:
          ImageEnView1.IO.Params.GIF_ImageIndex
            := iFrameIndex;
        ioTIFF:
          ImageEnView1.IO.Params.TIFF_ImageIndex
            := iFrameIndex;
      end;
```

TImageEnIO Links

```
      // Load the frame based in the index
      ImageEnView1.IO.LoadFromFile (iFilename);
    finally
      Screen.Cursor := crDefault;
    end;
  end;
```

Loading All The Frames

For example, TImageEnMView can load each frame in the image for display in a TImageEnView or the frames may be stored in memory with a TIEImageList. TIEImageList is a simple in memory list of TIEBitmap images.

TIEImageList To Store Frames and TListView to store frame descriptions and frame indexes

Important methods are shown in **blue.** This example uses a TListView, and a TImageEnView component and ImageEnIO created at runtime to load multi-frame images into a TIEImageList and a descriptions of the frames in TListView.

Helper functions

```
function AddThousandSeparator(const S: string;
const C: Char): string;
// Adds a specified thousand separator in the
// correct location in a string
var
  i: Integer;
// Loop through separator position
begin
  Result := S;
  i := Length (S) - 2;
  while i > 1 do
  begin
    Insert (C, Result, i);
    i := i - 3;
  end;
end;
```

TImageEnIO Links

```
function AddDefThousandSeparator(const S: string
): string;
// Adds the locale default thousand separator in
// the correct location in a string
begin
  Result := AddThousandSeparator(S,
     SysUtils.ThousandSeparator);
end;

function IntegerToString(const i: Int64): string;
// Adds the locale default thousand separator in
// the correct location in a number and returns a
// string
begin
  Result := AddThousandSeparator (IntToStr (i
            ), SysUtils.ThousandSeparator );
end;
```

Create a TIEImageList to hold the images in memory

```
procedure TForm1.FormCreate (Sender: TObject);
begin
  FIEImageList := TIEImageList.Create;
  ImageEnView1.Proc.AutoUndo := False;
  ImageEnView1.Proc.UndoLimit := 99;
  ImageEnView1.SetChessboardStyle(8);
end;
```

Free the TIEImageList

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
  // Free the multi-frame imagelist
    FIEImageList.Clear;
    FIEImageList.Free;
end;
```

Open a multi-frame or single frame ICO, GIF or TIF file

```
procedure TForm1.Open1Click (Sender: TObject);
var
  i: integer;
  iFrames: integer;
  iFileName: string;
  iExtension: string;
  iImageEnIO: TImageEnIO;
  iBitDepth: integer;
  iColor: string;
  iListItem: TListItem;
begin
  if OpenDialog1.Execute then
  begin
    // Close all frames
    FIEImageList.Clear;
    ListView1.Clear;
    iFileName := OpenDialog1.FileName;
    iExtension := Lowercase (ExtractFileExt
                  (iFileName));
    iFrames := IEGetFileFramesCount (iFileName);

    if iFrames > 1 then // Load multi-frame image
    begin
      ListView1.Items.BeginUpdate;
      try
        for i := 0 to iFrames - 1 do
        begin
          iImageEnIO := TImageEnIO.Create(nil);
          try
            // Important- ImageIndex determines
            // the frame that will be loaded
            if iExtension = '.ico' then
            begin
              iImageEnIO.Params.ICO_ImageIndex := i;
              ImageEnView1.IO.Params.ICO_Background:=
                TColor2TRGB(clBlack);
            end;
```

```
        if iExtension = '.gif' then
          iImageEnIO.Params.GIF_ImageIndex := i;
        if iExtension = '.tif' then
          iImageEnIO.Params.TIFF_ImageIndex := i;
        iImageEnIO.LoadFromFile(iFileName);
        // Add the image of the frame to
        // IEImageList
        FIEImageList.AppendImageRef
          (TIEBitmap.Create
          (iImageEnIO.IEBitmap), iFileName);
        // Show each frames information in a
        // TListView
        iListItem := ListView1.Items.Add;
        iListItem.Caption :=
          IntToStr(ListView1.Items.Count);
        iListItem.ImageIndex := 10;
        iListItem.SubItems.Add(iFileName);
        iListItem.SubItems.Add(IntegerToString(
          ImageEnView1.IEBitmap.Width) + ' x ' +
            IntegerToString(ImageEnView1.
            IEBitmap.Height) + ' pixels');
        iBitDepth :=
                 ImageEnIO.Params.SamplesPerPixel
               * iImageEnIO.Params.BitsPerSample;
        if iBitDepth = 32 then
          iColor := 'RGBA 32-Bit'
        else
          iColor := 'RGB ' + IntToStr(iBitDepth)
                    + '-bit';
        iListItem.SubItems.Add(iColor);
        ListView1.ItemIndex := 0;
      finally
        iImageEnIO.Free;
      end;
    end;
  finally
    ListView1.Items.EndUpdate;
  end;
```

```
end
else // Single-frame file
begin
  if iExtension = '.ico' then begin
   ImageEnView1.IO.Params.ICO_ImageIndex := 0;
   ImageEnView1.IO.Params.ICO_Background :=
      TColor2TRGB(clBlack);
  end;
  if iExtension = '.gif' then
    ImageEnView1.IO.Params.GIF_ImageIndex := 0;
    if iExtension = '.png' then
      ImageEnView1.IO.Params.PNG_Background :=
        TColor2TRGB(clBlack);
  ImageEnView1.IO.LoadFromFile ( iFileName );
  if ImageEnView1.IO.Aborting then
  begin
    MessageBeep( MB_ICONERROR );
    TaskDialog1.MainIcon := tdiError;
    TaskDialog1.Title := 'Error';
    TaskDialog1.Caption := 'Image Loading
                           Failed';
    TaskDialog1.Text := 'The image could not be
    loaded.';
    TaskDialog1.ExpandedText := 'The image could
                                not be
                                loaded and maybe
                                corrupt.';
   TaskDialog1.CommonButtons := [tcbOk];
   TaskDialog1.Execute;
   exit;
  end;
  //ImageEnView1 displays a frame depending on
  // the index set by the TListView's ItemIndex.
  ImageEnView1.EnableAlphaChannel := True;
  ImageEnView1.IO.Params.
    BMP_HandleTransparency := True;
```

```
      iBitDepth :=
        ImageEnView1.IO.Params.SamplesPerPixel
             *  ImageEnView1.IO.Params.BitsPerSample;
       if iBitDepth = 32 then
         iColor := 'RGBA 32-bit'
       else
         iColor := 'RGB ' + IntToStr(iBitDepth) + '-
                   bit';
       // Show each frames information in a TListView
       iListItem := ListView1.Items.Add;
       ListView1.Columns[0].Width := 50;
       iListItem.Caption :=
         IntToStr(ListView1.Items.Count);
       iListItem.SubItems.Add(
         ImageEnView1.IO.Params.FileName);
       iListItem.SubItems.Add (IntegerToString(
         ImageEnView1.Bitmap.Width) + ' pixels x ' +
         IntegerToString(ImageEnView1.Bitmap.Height )
            + ' pixels');
       iListItem.SubItems.Add(iColor);
       iListItem.ImageIndex := 10;
    end;
  end;
end;
```

TImageEnMView To Store Frames and frame descriptions and frame indexes

```
procedure TForm1.Open1Click(Sender: TObject);
var
  idx: integer;
  iBitDepth: integer;
  iColor: string;
begin
  if OpenDialog1.Execute then
  begin
    ImageEnMView1.StoreType := ietNormal;
    ImageEnMView1.Clear;
    ImageEnMView1.MIO.LoadFromFileAuto(
      OpenDialog1.FileName);
    for idx := 0 to ImageEnMView1.ImageCount - 1 do
    begin
      iBitDepth :=
        ImageEnMView1.MIO.Params[idx].SamplesPerPixel
       * ImageEnMView1.MIO.Params[idx].BitsPerSample;
      if iBitDepth = 32 then
        iColor := 'RGBA 32-Bit'
      else
        iColor := 'RGB ' + IntToStr(iBitDepth) + '-
                    bit';
      ImageEnMView1.ImageTopText[idx].Caption :=
        'Frame ' + IntToStr(idx + 1);
      ImageEnMView1.ImageInfoText[idx].Caption :=
        IntToStr(ImageEnMView1.ImageWidth[idx]) + ' x
        ' + IntToStr(ImageEnMView1.ImageHeight [idx]
             ) + ' ' + iColor;
      ImageEnMView1.ImageBottomText[idx].Caption :=
          ExtractFilename(OpenDialog1.FileName);
      Caption := OpenDialog1.FileName;
      ImageEnMView1.ImageBottomText[idx].Background
         := $00E8FFFF;
    end;
  end;
end;
```

Display the selected frame in TImageEnView.

```
procedure TForm1.ListView1SelectItem ( Sender: TObject; Item:
TListItem;  Selected:  Boolean );
begin
  if Selected then
  begin
     ImageEnView1.IEBitmap.Assign ( FIEImageList.Image
      [ Item.Index ] );
     ImageEnView1.Update;
     ImageEnView1.IEBitmap.UpdateFromTBitmap;
     Caption := ImageEnList - ' + FIEImageList.Filename [Item.Index];
  end;
end;
```

Saving Multi-frame files

Multi-frame images can be saved to a file with ImageEnView or ImageEnMView.

To save multi-frame images with TImageEnView:

```
procedure TForm1.FileSaveAs1Click(Sender: TObject);
var
  i: integer;
  j: integer;
  iFrames: array of TObject;
  iFileType: TIOFileType;
  iExtension: string;
  iBitDepth: integer;
  iColor: string;
begin
  if Assigned(PageControl1.ActivePage) then
  begin
    SavePictureDialog1.FileName := ExtractFileName(
      FFilePath);
    SavePictureDialog1.DefaultExt := ExtractFileExt(
      FFilePath);
    // Build the save picture dialog filter
    SavePictureDialog1.Filter := '';
    SavePictureDialog1.Filter :=
      SavePictureDialog1.Filter + 'TIFF Bitmap
        (TIF)|*.tif|';
    SavePictureDialog1.Filter :=
      SavePictureDialog1.Filter + 'CompuServe Bitmap
        (GIF)|*.gif|';
    SavePictureDialog1.Filter :=
      SavePictureDialog1.Filter + 'JPeg Bitmap
        (JPG)|*.jpg|';
    SavePictureDialog1.Filter :=
      SavePictureDialog1.Filter + 'PaintBrush
        (PCX)|*.pcx|';
    SavePictureDialog1.Filter :=
       SavePictureDialog1.Filter + 'Windows Bitmap
        (BMP)|*.bmp|';
    SavePictureDialog1.Filter :=
      SavePictureDialog1.Filter + 'Windows Icon
        (ICO)|*.ico|';
    SavePictureDialog1.Filter :=
      SavePictureDialog1.Filter + 'Portable Network
       Graphics (PNG)|*.png|';
    SavePictureDialog1.Filter :=
       SavePictureDialog1.Filter + 'Windows Metafile
         (WMF)|*.wmf|';
    SavePictureDialog1.Filter :=
```

```
    SavePictureDialog1.Filter + 'Windows Enhanced
      Metafile (EMF)|*.emf|';
  SavePictureDialog1.Filter :=
    SavePictureDialog1.Filter + 'Targa Bitmap
      (TGA)|*.tga|';
  SavePictureDialog1.Filter :=
    SavePictureDialog1.Filter + 'Portable Pixmap,
      GreyMap, BitMap (PXM)|*.pxm|';
  SavePictureDialog1.Filter :=
    SavePictureDialog1.Filter + 'Jpeg2000
      (JP2)|*.jp2|';
  SavePictureDialog1.Filter :=
  SavePictureDialog1.Filter + 'Jpeg2000 (J2K)|*.j2k|';
  SavePictureDialog1.Filter :=
    SavePictureDialog1.Filter + 'Wireless Bitmap
      (WBMP)|*.wbmp|';
  SavePictureDialog1.Filter :=
    SavePictureDialog1.Filter + 'Multipage PCX
      (DCX)|*.dcx|';
  SavePictureDialog1.Filter :=
     SavePictureDialog1.Filter + 'Microsoft HD
        Photo (HDP)|*.hdp|';
  SavePictureDialog1.Filter :=
    SavePictureDialog1.Filter + 'Microsoft Media
       Photo (WDP)|*.wdp|';
  SavePictureDialog1.DefaultExt := Lowercase(
     ExtractFileExt(FFilePath));
  ImageEnView := TImageEnView (
     cxPageControl1.ActivePage.Controls [0]);
  if Assigned(ImageEnView) then
  begin
    SavePictureDialog1.FilterIndex :=
      IEFileTypeToGraphicFilterIndex(
        ImageEnView.IO.Params.FileType);
  end;
  if SavePictureDialog1.Execute then
  begin
    if SavePictureDialog1.FileName <> '' then
    begin
      Screen.Cursor := crHourGlass;
      try
        FFilePath := SavePictureDialog1.FileName;
```

```
iExtension :=
  Lowercase(ExtractFileExt(FFilePath));
ImageEnView := TImageEnView(
  PageControl1.ActivePage.Controls[0]);
  if Assigned (ImageEnView) then
  begin
    // Set the filetype based on the extension
    if iExtension = '.ico' then
     ImageEnView.IO.Params.FileType := ioICO
    else if iExtension = '.bmp' then
     ImageEnView.IO.Params.FileType := ioBMP
    else if iExtension = '.png' then
      ImageEnView.IO.Params.FileType := ioPNG
    else if iExtension = '.tif' then
      ImageEnView.IO.Params.FileType := ioTIFF
    else if iExtension = '.wmf' then
      ImageEnView.IO.Params.FileType := ioWMF
    else if iExtension = '.emf' then
      ImageEnView.IO.Params.FileType := ioEMF
    else if iExtension = '.tga' then
      ImageEnView.IO.Params.FileType := ioTGA
    else if iExtension = '.jp2' then
      ImageEnView.IO.Params.FileType := ioJP2
    else if iExtension = '.j2k' then
      ImageEnView.IO.Params.FileType := ioJ2K
    else if iExtension = '.hdp' then
      ImageEnView.IO.Params.FileType := ioHDP
    else if iExtension = '.gif' then
      ImageEnView.IO.Params.FileType := ioGIF
    else if iExtension = '.pcx' then
      ImageEnView.IO.Params.FileType := ioPCX
```

```
else
  ImageEnView.IO.Params.FileType := ioUnknown;
iFileType := ImageEnView.IO.Params.FileType;
if iFileType = ioICO then // Icon
begin
  SetLength (iFrames, PageControl1.PageCount);
  Progressbar1.Max := PageControl1.PageCount-1;
  for i := 0 to PageControl1.PageCount - 1 do
  begin
    ImageEnView := TImageEnView(
      PageControl1.Pages[i].Controls[[0]);
    if Assigned(ImageEnView) then
    begin
       iFrames[i] := ImageEnView;
     ImageEnView.IO.Params.ICO_ImageIndex := i;
     Progressbar1.Position := i;
     Application.ProcessMessages;
      // Saving is too fast... slow it down
      Sleep(100);
    end;
  end;
  // Save the icon
  IEWriteICOImages(FFilePath, iFrames);
  if Assigned(ImageEnView) then
  begin
     ImageEnView.Bitmap.Modified := False;
     ImageEnView.ImageChange;
  end;
  Progressbar1.Position := 0;
 Application.ProcessMessages;
end
else // Not icon
begin
   if Assigned(ImageEnView) then
   begin
    ImageEnView.IO.PreviewsParams := [
      ioppDefaultLockPreview
    ImageEnView.IO.SimplifiedParamsDialogs :=
      False;
    iegDefaultPreviewsZoomFilter := rfNone;
```

```
          if ImageEnView.IO.DoPreviews([ppAUTO]) then
             begin
               ImageEnView.IO.SaveToFile (FFilePath);
               iBitDepth :=
                 ImageEnView.IO.Params.SamplesPerPixel *
                 ImageEnView.IO.Params.BitsPerSample;
               if iBitDepth = 32 then
                 iColor := 'RGBA 32-bit'
               else
                 iColor := 'RGB ' + IntToStr(iBitDepth)
                   + '-bit';
               Caption := FFilePath;
              PageControl1.ActivePage.Caption :=
               ExtractFilename(FFilePath) +
                 ' ' + IntegerToString(
                  ImageEnView.Bitmap.Width) + '
                     pixels x ' + IntegerToString
                      (ImageEnView.Bitmap.Height) + '
                         pixels' + ' ' + iColor;
               StatusBar1.Panels [0].Text :=
                  ExtractFilePath(FFilePath);
               StatusBar1.Panels [1].Text :=
                 ExtractFileName (FFilePath);
               ImageEnView.Bitmap.Modified := False;
            end;
           end;
           Progressbar1.Position := 0;
        end;
      end;
    finally
      Screen.Cursor := crDefault;
    end;
  end
  else if ImageEnView.IO.Params.FileType <> ioICO
  then
    // Save as single frame
    begin
      ProgressBar1.Max := 100;
      ImageEnView.IO.SimplifiedParamsDialogs :=
         False;
      ImageEnView.IO.PreviewsParams := [
         ioppDefaultLockPreview];
```

```
          if ImageEnView.IO.DoPreviews([ppAUTO])
          then
            begin
              ImageEnView.IO.SaveToFile(FFilePath);
          ImageEnView.Bitmap.Modified := False;
          ImageEnView.ImageChange;
          iBitDepth :=
            ImageEnView.IO.Params.SamplesPerPixel *
              ImageEnView.IO.Params.BitsPerSample;
          if iBitDepth = 32 then
            iColor := 'RGBA 32-bit'
          else
            iColor := 'RGB ' + IntToStr (iBitDepth) +
              '-bit';
          Caption :=  FFilePath;
          PageControl1.ActivePage.Caption :=
             ExtractFilename(FFilePath)
               + ' ' +  IntegerToString(
          ImageEnView.Bitmap.Width) + ' pixels x '
            + IntegerToString(
         ImageEnView.Bitmap.Height) + ' pixels' + ' '
            + iColor;
          StatusBar1.Panels[0].Text := ExtractFilePath
           (FFilePath);
          StatusBar1.Panels[1].Text :=
             ExtractFileName(FFilePath);
        end;
      end;
    end;
  end;
end;
```

To save multi-frame files with TImageEnMView:

```
procedure TForm1.Save1Click(Sender: TObject);
begin
  if SavePictureDialog1.Execute then
      ImageEnMView1.MIO.SaveToFile(
         SavePictureDialog1.FileName);
end;
```

Image capture (Twain and WIA)

Unfortunately I am not an expert with Twain and WIA nor do I have a lot of equipment on which to test so I probably will not say much about this other than most all of the questions I have seen about twain in the ImageEn forum show that the majority of the problems using ImageEn with twain and WIA is the result of problems **caused by the individual scanner and device makers twain software or code and not by ImageEn itself**. The majority of problems using Twain or WIA is not caused by ImageEn, rather it is usually the scanner or devices software driver that causes the problem. Some companies twain software is very good while others is not so good. During my own personal use of Twain and WIA I have found that there can even be significant differences with successful integration of twain or WIA with ImageEn even from a single manufactures device. What I mean by this is sometimes the manufactures Twain driver may work better than its WIA driver or vice versa. What this means is unless you need to support both, use the driver that that seems to function the best and suites your needs.

Unit ImageEnIO

TImageEnIO

Methods and Properties

Connected component

AttachedBitmap	AttachedIEBitmap
AttachedImageEn	AttachedTImage

Generic Input/Output

Aborting	AssignParams
Background	Bitmap
ChangeBackground	Create
DefaultDitherMethod	FilteredAdjustDPI
LoadFromBuffer	LoadFromFileAuto
LoadFromFile	LoadFromResource
LoadFromStream	LoadFromURL
Params	ParamsFromBuffer
ParamsFromFile	ParamsFromStreamFormat
ProxyAddress	ProxyPassword
SaveToFile	SaveToStream
StreamHeaders	Update
Loading Multi-frame	Saving Multi-frame

Unit ImageEnIO
TImageEnIO

TWain/WIA

AcquireClose	AcquireOpen
Acquire	SelectAcquireSource
TWainParams	WIAParams

Asynchronous Input/Output

AsyncMode	AsyncRunning
ThreadsCount	WaitThreads

DirectShow Capture

DShowParams

Dialogs

DialogsMeasureUnit	DoPreviews
ExecuteOpenDialog	ExecuteSaveDialog
MsgLanguage	PreviewFont
PreviewsParams	SimplifiedParamsDialogs

Printing

DoPrintPreviewDialog	PreviewPrintImage
PrintImagePos	PrintImage
PrintingFilterOnSubsampling	PrintPreviewParams

Unit ImageEnIO
TImageEnIO

JPEG

InjectJpegEXIFStream	InjectJpegEXIF
InjectJpegIPTCStream	InjectJpegIPTC
LoadFromFileJpeg	LoadFromStreamJpeg
SaveToFileJpeg	SaveToStreamJpeg

JPEG 2000

LoadFromFileJ2K	LoadFromFileJP2	LoadFromStreamJ2K
LoadFromStreamJP2	SaveToFileJ2K	SaveToFileJP2
SaveToStreamJ2K	SaveToStreamJP2	

GIF

InsertToFileGIF	LoadFromFileGIF	LoadFromStreamGIF
SaveToFileGIF	SaveToStreamGIF	ReplaceFileGIF

Adobe PSD

LoadFromFilePSD	LoadFromStreamPSD	SaveToFilePSD
SaveToStreamPSD		

Unit ImageEnIO
TImageEnIO

HDP - Microsoft HD Photo

LoadFromFileHDP LoadFromStreamHDP SaveToFileHDP

SaveToStreamHDP

TIFF

InjectTIFFEXIF InsertToFileTIFF InsertToStreamTIFF

LoadFromFileTIFF LoadFromStreamTIFF ReplaceFileTIFF

ReplaceStreamTIFF SaveToFileTIFF SaveToStreamTIFF

BMP

LoadFromFileBMP LoadFromStreamBMP SaveToFileBMP

SaveToStreamBMP

PNG

LoadFromFilePNG LoadFromStreamPNG SaveToFilePNG

SaveToStreamPNG

DCX

LoadFromFileDCX LoadFromStreamDCX SaveToStreamDCX

SaveToFileDCX InsertToFileDCX

Unit ImageEnIO
TImageEnIO

CUR

LoadFromFileCUR LoadFromStreamCUR

ICO

LoadFromFileICO LoadFromStreamICO SaveToFileICO

SaveToStreamICO

PCX

LoadFromFilePCX LoadFromStreamPCX SaveToFilePCX

SaveToStreamPCX

WMF, EMF

ImportMetafile MergeMetaFile

TGA

LoadFromFileTGA LoadFromStreamTGA SaveToFileTGA

SaveToStreamTGA

PXM (PPM, PBM,PGM)

LoadFromFilePXM	LoadFromStreamPXM	SaveToFilePXM
SaveToStreamPXM		

AVI

CloseAVIFile	CreateAVIFile	IsOpenAVI
LoadFromAVI	OpenAVIFile	SaveToAVI

WBMP

LoadFromFileWBMP	LoadFromStreamWBMP	SaveToFileWBMP
SaveToStreamWBMP		

PostScript (PS)

ClosePSFile	CreatePSFile	SaveToFilePS
SaveToPS	SaveToStreamPS	

Adobe PDF

ClosePDFFile	CreatePDFFile	SaveToFilePDF
SaveToPDF	SaveToStreamPDF	

Unit ImageEnIO
TImageEnIO

RAW Camera

LoadFromFileRAW LoadFromStreamRAW LoadJpegFromFileCRW

MediaFiles (AVI, MPEG, WMV..)

CloseMediaFile IsOpenMediaFile LoadFromMediaFile

OpenMediaFile

Real RAW (not Camera RAW)

LoadFromFileBMPRAW LoadFromStreamBMPRAW SaveToFileBMPRAW

SaveToStreamBMPRAW

DICOM Medical Imaging

LoadFromFileDICOM LoadFromStreamDICOM

Events

OnAcquireBitmap OnDoPreviews OnFinishWork

OnIOPreview OnProgress

Methods and Properties

Connected component

Unit ImageEnIO

TImageEnIO

DECLARATION

```
property AttachedBitmap: TBitmap;
```

DESCRIPTION

Use this property if you want to attach TImageEnIO to a normal TBitmap object. This property is mutually exclusive with AttachedTImage and AttachedImageEn.

Example

```
var
iBitmap: TBitmap;
begin
  iBitmap := TBitmap.Create;
  ImageEnIO1.AttachedBitmap := iBitmap;
  ImageEnIO1.LoadFromFile('alfa.tif');
...
end;
```

ImageEnIO

DECLARATION

```
property AttachedIEBitmap: TIEBitmap
```

DESCRIPTION

AttachedIEBitmap contains the attached TIEBitmap object (some value of IEBitmap property).

Example

```
// multithread saving
var
iBitmap: TIEBitmap;
iImageEnIO: TImageEnIO;
begin
  iBitmap := TIEBitmap.Create;
  iImageEnIO := TImageEnIO.CreateFromBitmap(bmp);
  iImageEnIO.LoadFromFile('input.bmp');
  iImageEnIO.AsyncMode:=True;
  // this makes a thread for each input/output
  // task
  iImageEnIO.SaveToFile('i1.jpg'); // thread-1
  iImageEnIO.SaveToFile('i2.jpg'); // thread-2
  iImageEnIO.SaveToFile('i3.jpg'); // thread-3
  // free method waits for all tasks
  iImageEnIO.Free;
end;
  Bmp.Free;
end;
```

ImageEnIO

DECLARATION

```
property AttachedImageEn: TIEView;
```

DESCRIPTION

Use this property if you want to attach TImageEnIO to a TImageEn, TImageEnView or TImageEnDBView (or any other inherited object) components. This property is mutually exclusive with AttachedTImage and AttachedBitmap. TIEView is the base class of TImageEnView and TImageEnMView.

DECLARATION

```
property AttachedTImage: TImage;
```

DESCRIPTION

Use this property if you want to attach TImageEnIO to a TImage (any other inherited object) component. This property is mutually exclusive with AttachedImageEn and AttachedBitmap.

Example

```
ImageEnIO1.AttachedTImage := Image1;
```


Generic Input/Output

DECLARATION

```
property Aborting: Boolean;
```

Unit ImageEnIO
TImageEnIO

DESCRIPTION

Applications can abort save/load processing by assigning True to the Aborting property. If the value is set to true during loading, the image will be truncated. If it is set to True during saving, the file will be closed and truncated (will be unreadable). You can also read the Aborting property to know when aborting is in progress.

ImageEnIO

Example

```
/ Begin to load image
procedure TForm1.Button1Click(Sender: TObject);
begin
  ImageEnView1.IO.LoadFromFile('xyx.bmp');
end;

// Progress of load
procedure TForm1.ImageEnIO1Progress(Sender:
TObject; per: Integer);
begin
  ProgressBar1.Position := per;
  Application.ProcessMessages; // <- important!
end;

// STOP! button
procedure TForm1.Button1Click(Sender: TObject);
begin
  ImageEnView1.IO.Aborting := True;
end;
```

DECLARATION

```
procedure AssignParams(Source: TObject);
```

DESCRIPTION

Assigns file format parameters from another TImageEnIO component or TIOParamsVals object.

ImageEnIO

DECLARATION

```
property AutoAdjustDPI: Boolean;
```

DESCRIPTION

When AutoAdjustDPI is True and last loaded/scanned image has horizontal DPI not equal to vertical DPI, ImageEn resizes the image making DPIX=DPIY. The default is False.

DECLARATION

```
property Background: TColor;
```

DESCRIPTION

This property sets the background color. The background color is the color shown in unoccupied area when the current image is less of control size. When TImageEnIO is attached to TImageEnView the TImageEnIO.Background is equal to Background.

ImageEnIO

DECLARATION

```
property Bitmap: TBitmap;
```

DESCRIPTION

References the bitmap contained in TImageEnIO, TImageEnView or TImageEnMView.

Read-only

Example

```
// This use standard Delphi LoadFromFile
ImageEnView1.IO.Bitmap.LoadFromFile('my.bmp');
ImageEnView1.IO.Bitmap.PixelFormat := pf24bit;
ImageEnView1.Update;
```

ImageEnIO

DECLARATION

```
procedure CaptureFromScreen(Source: TIECSSource;
MouseCursor: TCursor);
```

DESCRIPTION

CaptureFromScreen captures current desktop or active window. MouseCursor defines the cursor to draw because ImageEn cannot get it from Windows. Use -1 for none.

Example

```
// Saves current desktop in 'screen.png'
ImageEnView1.io.CaptureFromScreen(iecsScreen,-1);
ImageEnView1.io.SaveToFile('screen.png');
```

See Apprehend for additional screen capture functions.
http://www.frontiernet.net/~w2m/index.html

DECLARATION

```
TIECSSource = (iecsScreen, iecsForegroundWindow,
iecsForegroundWindowClient);
```

DESCRIPTION

iecsScreen : capture current desktop
iecsForegroundWindow : capture foreground window
iecsForegroundWindowClient : capture foreground window client area

ImageEnIO

DECLARATION

```
property ChangeBackground: Boolean;
```

DESCRIPTION

Set ChangeBackground to True to change the TImageEnView control background color to the image background color (the default is False). Not all file formats contains background color information.

DECLARATION

```
constructor Create(Owner: TComponent);
```

DESCRIPTION

Create creates a new instance. You can set Owner=nil to create a component without owner.

ImageEnIO

DECLARATION

TImageEnIO

```
constructor CreateFromBitmap(Bitmap: TIEBitmap);
constructor CreateFromBitmap(Bitmap: TBitmap);
```

DESCRIPTION

`CreateFromBitmap` creates a new instance assigning property AttachedIEBitmap or AttachedBitmap.

Example

```
with TImageEnIO.CreateFromBitmap(myBitmap) do
begin
  LoadFromFile('input.jpg');
  Free;
end;
```

ImageEnIO

DECLARATION

```
property DefaultDitherMethod: TIEDitherMethod;
```

DESCRIPTION

DefaultDitherMethod specifies the default dithering method to apply when a color image needs to be converted to black/white. IeThreshold is the default.

DECLARATION

TIEDitherMethod = (ieOrdered, ieThreshold);

DECLARATION

```
property FilteredAdjustDPI: Boolean;
```

DESCRIPTION

This property is valid when AutoAdjustDPI is true. If set to true, ImageEn applies a resampling filter to the image to enhance quality. It can slow down the loading process.

ImageEnIO

DECLARATION

```
property IEBitmap: TIEBitmap
```

DESCRIPTION

IEBitmap contains the attached TIEBitmap object (some value of AttachedIEBitmap property).

DECLARATION

```
procedure LoadFromBuffer(Buffer: pointer;
BufferSize: integer; Format: TIOFileType =
ioUnknown);
```

DESCRIPTION

LoadFromBuffer loads an image from the specified buffer.

Parameter	Description
Buffer	The buffer pointer
BufferSize	The buffer length in bytes.
Format	Specifies the expected file format. If Format is ioUnknown, then try to find the format automatically

Example

```
ImageEnView1.IO.LoadFromBuffer(mybuffer,
mybufferlength, ioJPEG);
```

DECLARATION

```
procedure LoadFromFileAuto(const FileName:
WideString); dynamic;
```

DESCRIPTION

LoadFromFileAuto loads an image from file. To detect file format it doesn't look at the file extension, but at the file content. This method can load LYR (TImageEnView layers), IEV (TImageEnVect objects) and LYR+IEV formats when AttachedImageEn is TImageEnView or TImageEnVect.

Example

```
ImageEnView1.IO.LoadFromFileAuto('input.dat');
ImageEnView1.IO.LoadFromFileAuto('input.tif');
// a tiff or a RAW?
```

Unit ImageEnIO
TImageEnIO

DECLARATION

```
procedure LoadFromFileFormat(const FileName:
WideString; FileFormat: TIOFileType);
```

DESCRIPTION

LoadFromFileFormat loads an image from file. LoadFromFileFormat detects the file format from FileFormat parameter. This method can load LYR (TImageEnView layers), IEV (TImageEnVect objects) and LYR+IEV formats when AttachedImageEn is TImageEnView or TImageEnVect.

If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name.

Example

```
// load ximage.dat as a Jpeg file
ImageEnView1.IO.LoadFromFileFormat('ximage.dat',
ioJPEG)

// load layers and vectorial objects
ImageEnVect1.IO.LoadFromFileFormat('file.all',
ioALL);
```

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure LoadFromFile(const FileName:
WideString);
```

DESCRIPTION

LoadFromFile loads an image from the specified file. It recognizes the image format from the filename extension. The source can be also an URL, if it has the form 'http://'. This method can load LYR (TImageEnView layers), IEV (TImageEnVect objects) and LYR+IEV formats when AttachedImageEn is TImageEnView or TImageEnVect.

LoadFromFile detects file format from the extension of the specified file. If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension.

Example

```
// loads the second image in the document.tif
ImageEnView1.IO.Params.TIFF_ImageIndex:=1;
ImageEnView1.IO.LoadFromFile('document.tif');
```

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure LoadFromResource(const ModulePath:
WideString; const ResourceType: string; const
ResourceName: string; Format: TIOFileType);
```

DESCRIPTION

LoadFromResource loads the specified image resource from PE files like EXE, DLL, OCX, ICL, BPL, etc.

Parameter	Description
ModulePath	Specifies the path and filename of PE module.
ResourceType	Resource type as string (ie 'Bitmap', 'Cursor').
ResourceName	Resource name as string (ie 'INTRESOURCE: 100', 'Hand').
	Specifies the expected file format. If Format is ioUnknown, then try to find the format automatically. IoUnknown should fail for BMP, ICO and CUR resources.

See also TIEResourceExtractor.

ImageEnIO

Example

```
// loads resource 143, in "Bitmap"s, from
"explorer.exe" (should be a little Windows logo)
ImageEnView1.IO.LoadFromResource('explorer.exe',
'Bitmap', 'INTRESOURCE:143', ioBMP);
```

DECLARATION

```
procedure LoadFromStreamFormat(Stream: TStream;
FileFormat: TIOFileType);
```

DESCRIPTION

LoadFromStreamFormat loads an image from stream. LoadFromStreamFormat detects file format from FileFormat parameter. This method can also load LYR (TImageEnView layers), IEV (TImageEnVect objects) and LYR+IEV formats when AttachedImageEn is TImageEnView or TImageEnVect.

If the file does not represent a valid image format, the Aborting property will be true.

If FileFormat is ioUnknown, then LoadFromStream is called (auto detection of the file format).

Example

```
// load a jpeg from the stream
ImageEnView1.IO.LoadFromStreamFormat(iStream,
ioJPEG)
```

Unit ImageEnIO
TImageEnIO

DECLARATION

```
procedure LoadFromStream(Stream: TStream);
```

DESCRIPTION

LoadFromStream loads the image from a stream by calling FindStreamFormat to select file format. This method can load LYR (TImageEnView layers), IEV (TImageEnVect objects) and LYR+IEV formats when AttachedImageEn is TImageEnView or TImageEnVect.

If the stream does not represent a valid image format, the Aborting property is true. If the StreamHeaders property is true, the stream must have a special header (saved with SaveToStream).

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure LoadFromURL(const URL:WideString);
```

DESCRIPTION

LoadFromURL loads the image from the network using the HTTP or FTP protocol, specifying the URL. This function doesn't support password authentication for HTTP, while it is necessary for FTP (also connecting to anonymous server).

URL must have the syntax:
'http://domain[:port]/resource'
'https://domain[:port]/resource'
'ftp://user:password@domain[:port]/resource'

It is possible to set proxy parameters using ProxyAddress, ProxyUser and ProxyPassword properties.

Example

```
// load from standard port 80
ImageEnView1.IO.LoadFromURL('http://www.imageen.com/image.jpg');

// load from port 8080
ImageEnView1.IO.LoadFromURL('http://www.imageen.com:8080/image.jpg');

// load from FTP
ImageEnView1.IO.LoadFromURL('ftp://space:shuttle@ftp.imageen.com/Pictures/test.jpg')
```

Unit ImageEnIO
TImageEnIO

DECLARATION

```
property NativePixelFormat: Boolean;
```

DESCRIPTION

By setting this property to true, you disable the conversion of paletted images, gray scale and all other formats to 24 bit or 1 bit. By default, ImageEn converts all paletted, gray scale images and other formats to 24 bit (true color). Only black/white images are stored in the original format with 1 bit per pixel. LegacyBitmap must be False.

Note that setting NativePixelFormat = True, you will not be able to execute some image processing operations.

For example, if you have a 16 bit gray scale TIFF and you want to work with 16 bit gray scale pixels, you have to write this before load the image:

```
ImageEnView1.LegacyBitmap:=False; // do not use
TBitmap
ImageEnView1.IO.NativePixelFormat:=true; // leave
original pixel format
```

Now you can read pixels using:

```
word = ImageEnView1.IEBitmap. Pixels_ie16g[x,y];
```

Unit ImageEnIO

TImageEnIO

DECLARATION

```
property Params: TIOParamsVals;
```

DESCRIPTION

This property contains a TIOParamsVals object. This property contains some parameters (as bits per sample, type of compression, etc) of each image file format. The parameters are updated when you load files or streams. Also you can modify these parameters before saving to files or streams.

Read-only

Example

```
// Sets colormapped 256 colors
ImageEnView1.IO.Params.BitsPerSample:=8;
ImageEnView1.IO.Params.SamplesPerPixel:=1;
// Sets OS2.x compressed BMP
ImageEnView1.IO.Params.BMP_Version:=ioBMP_BMOS2V2;
ImageEnView1.IO.Params.BMP_Compression:=ioBMP_RLE;
// Saves
ImageEnView1.IO.SaveToFile('lady.bmp');
```

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure ParamsFromBuffer(Buffer: pointer;
BufferSize: integer; Format: TIOFileType =
ioUnknown);
```

DESCRIPTION

Loads image parameters (but not the actual image) from the specified buffer.

Parameter	Description
Buffer	The buffer pointer
BufferSize	The buffer length in bytes.
Format	Specifies the expected file format. If Format is ioUnknown, then try to find the format automatically

See also

LoadFromBuffer

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure ParamsFromFileFormat(const FileName:
WideString; format: TIOFileType);
```

DESCRIPTION

The ParamsFromFileFormat method fills Params when reading an image from Stream, but don't load the image into the component.

For ParamsFromFileFormat, FileName is the file name with full path. Format is the file format you know the stream or file contains (you cannot specify ioUnknown).

DECLARATION

```
procedure ParamsFromFile(const FileName:
WideString);
```

DESCRIPTION

ParamsFromFile reads the image properties (Params) without loading the image (and without changing the current image).

ParamsFromFile (and ParamsFromStream) auto recognize image type from file header and file name extension. If the image is unknown file type then the Params.FileType has ioUnknown value.

See also
ParamsFromStreamFormat
ParamsFromFileFormat

ImageEnIO

Unit ImageEnIO
TImageEnIO

DECLARATION

```
procedure ParamsFromStreamFormat(Stream: TStream;
format: TIOFileType);
```

DESCRIPTION

The ParamsFromStreamFormat method fills Params when reading an image from Stream, but don't load the image into the component.

Stream is a TStream that contains the image. Format is the file format you know the stream or file contains (you cannot specify ioUnknown).

DECLARATION

```
procedure ParamsFromStream(Stream: TStream);
```

DESCRIPTION

ParamsFromStream reads the image properties (Params) without loading the image (and without changing the current image).

ParamsFromStream auto-recognizes image type from file header and file name extension. If the image is unknown file type then Params.FileType has ioUnknown value.

DECLARATION

```
property ProxyAddress: WideString;
```

DESCRIPTION

ProxyAddress specifies the proxy address and port when using LoadFromURL method. The syntax is: ‘domain:port’.

DECLARATION

```
property ProxyPassword: WideString;
```

DESCRIPTION

ProxyPassword specifies the proxy password when using LoadFromURL method.

DECLARATION

```
property ProxyUser: WideString;
```

DESCRIPTION

ProxyUser specifies the proxy userid when using LoadFromURL method.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure SaveToFile(const FileName: WideString);
```

DESCRIPTION

SaveToFile saves the current image to Jpeg, Jpeg2000, PNG, TIFF, BMP, WBMP, PS, PDF, PCX, DCX, TGA, PXM, ICO, HDP, GIF and all other supported formats.

SaveToFile detects file format from the extension of the specified file. FileName is the file name with extension.

DECLARATION

```
procedure SaveToStream(Stream: TStream; FileType:
TIOFileType);
```

DESCRIPTION

SaveToStream saves the image to a stream by calling FileType to specify the file format.

If StreamHeaders property is true adds an additional special header, needed for multi-image streams.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure SaveToText(const FileName: WideString;
ImageFormat: TIOFileType; TextFormat:
TIETextFormat);
```

DESCRIPTION

SaveToText saves current image in the specified text format.

Parameters:

Parameter	Description
FileName	Output file name.
ImageFormat	Wanted file format.
TextFormat	Output text format. See allowed values in TIETextFormat.

ImageEnIO

Unit ImageEnIO

TImageEnIO

DECLARATION

```
function Seek(Page: TIEIOSeekDestination;
FileName: WideString = ''):integer;
```

DESCRIPTION

Seek loads the specified page from the last loaded multipage file, including TIFF, GIF, AVI and other video types. This method uses LoadFromFile to load the file (hence the file name extension must be provided), unless it is part of TImageEnDBView or TImageEnDBVect where LoadPicture is called instead.

Parameter	Description
Page	Page to load.
FileName	Optional file name. If empty will use the last loaded file name.

Returns loaded page index.

Example

```
    // load first page
ImageEnView1.IO.Seek(ieioSeekFirst,
'multipage.tiff');

// load next page
ImageEnView1.IO.Seek(ieioSeekNext);
```

Unit ImageEnIO

TImageEnIO

DECLARATION

```
property StreamHeaders: Boolean;
```

DESCRIPTION

If StreamHeaders is true, each SaveToStreamXXX method adds an additional special header as needed for multi-image streams. When a special header is added, the images saved with SaveToStreamXXX aren't compatible with LoadFromFileXXXX methods.

DECLARATION

```
procedure Update;
```

DESCRIPTION

If TImageEnIO is attached to TImageEnView (or inherited) object, Update calls relative Update method. If TImageEnIO is attached to TBitmap object, Update sets Modified property to True.

Loading Multi-frame Files

Saving Multi-frame Files

Twain/WIA

DECLARATION

```
procedure AcquireClose;
```

DESCRIPTION

AcquireClose closes a connection opened with AcquireOpen. It is useful to do a modeless acquisition. Whenever ImageEn gets an image the OnAcquireBitmap event occurs.

DECLARATION

```
function AcquireOpen: Boolean;
```

DESCRIPTION

AcquireOpen opens a connection to the selected scanner. It is useful to do a modeless acquisition. AcquireOpen returns False if it fails. Whenever ImageEn gets an image, the OnAcquireBitmap event occurs.

See also
AcquireClose.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
function Acquire(api: TIEAcquireApi): Boolean;
```

DESCRIPTION

The Acquire method performs image acquisition from Twain or WIA compatible scanners. Returns True if the acquisition has succeeded, False otherwise. It is suggested to call SetDefaultParams before call Acquire to reset parameters if you have already loaded an image from file.

DECLARATION

```
function SelectAcquireSource(api: TIEAcquireApi):
Boolean;
```

DESCRIPTION

SelectAcquireSource shows a dialog where the user can select a Twain or WIA device.

SelectAcquireSource returns false if user press “Cancel” button.

Unit ImageEnIO
TImageEnIO

DECLARATION

```
property TWainParams: TIETWainParams;
```

DESCRIPTION

Use the TWainParams property to control scanner acquisition. You can enable/disable standard user interface, set pixeltype (Grayscale, RGB...), DPI, or select the acquire scanner without user interaction.

Look at TIETWainParams object for more details.

Example

```
// Change scan resolution to 300 on your scanner
ImageEnView1.IO.TWainParams.XResolution.CurrentValue:=300;
ImageEnView1.IO.TWainParams.YResolution.CurrentValue:=300;
ImageEnView1.IO.Acquire;
```

DECLARATION

```
property WIAParams: TIEWia;
```

DESCRIPTION

WIAParams allows you to set/get parameters, show dialogs and control WIA devices. Look at TIEWia for more info.

Unit ImageEnIO

TImageEnIO

Asynchronous Input/Output

DECLARATION

```
property AsyncMode: Boolean;
```

DESCRIPTION

Set AsyncMode to True to enable asynchronous input/output operations. When asynchronous mode is enabled, each input/output method creates a new thread that executes the task then the called method returns without waiting for the end of the task. Applicable only when TImageEnIO is attached to a TIEBitmap object.

TImageEnIO

DECLARATION

```
property AsyncRunning: integer;
```

DESCRIPTION

AsyncRunning returns how many threads are running.
See also AsyncMode.

DECLARATION

```
property ThreadsCount: integer;
```

DESCRIPTION

ThreadsCount returns the count of currently active threads. Valid only if AsyncMode is true.

DECLARATION

```
procedure WaitThreads(Aborts: Boolean = false);
```

DESCRIPTION

WaitThreads waits until all threads complete. If Aborts is True, all threads will abort.

DirectShow Capture

Declaration

```
property DShowParams: TIEDirectShow;
```

Description

DShowParams is a TIEDirectShow instance which allows control of some Direct Show features, such as video capture, audio capture, multimedia files capture as well video rendering, and multimedia file writing.

Dialogs

DECLARATION

```
property DialogsMeasureUnit:
TIEDialogsMeasureUnit
```

DESCRIPTION

The DialogsMeasureUnit property specifies the measurement unit used in the print preview dialog (see DoPrintPreviewDialog).

DECLARATION

```
function DoPreviews(pp: TPreviewParams): Boolean;
```

DESCRIPTION

This function executes the Previews dialog. This dialog gets/sets the parameters of image file formats.

pp is the set of the image formats parameters to show in the dialog.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
function ExecuteOpenDialog(const InitialDir:
WideString; const InitialFileName: WideString;
AlwaysAnimate: Boolean; FilterIndex: integer;
const ExtendedFilters: WideString; const Title:
WideString=''; const Filter: WideString=''):
WideString;
```

DESCRIPTION

The ExecuteOpenDialog shows and executes the open dialog. It encapsulates the TOpenImageEnDialog component. InitialDir is the starting directory. InitialFileName is the default file name with extension. AlwaysAnimate if true auto-animate GIF and AVI FilterIndex specifies what file format to select for default. Allowed values:

1 : Common graphics formats
2 : All Graphics formats
3 : TIFF Bitmap (TIF;TIFF;FAX)
4 : GIF (GIF)
5 : JPEG Bitmap (JPG;JPEG;JPE)
6 : PaintBrush (PCX)
7 : Windows Bitmap (BMP;DIB;RLE)
8 : Windows Icon (ICO)
9 : Windows Cursor (CUR)
10 : Portable Network Graphics (PNG)
11 : DICOM medical imaging (DCM)
12 : Windows Metafile (WMF)
13 : Enhanced Windows Metafile (EMF)

14 : Targa Bitmap (TGA;TARGA;VDA;ICB;VST;PIX)
15 : Portable Pixmap, GreyMap, BitMap (PXM;PPM;PGM;PBM)
16 : Wireless Bitmap (WBMP)
17 : Jpeg2000 (JP2)
18 : Jpeg2000 Code Stream (J2K;JPC;J2C)
19 : Multipage PCX (DCX)
20 : Camera RAW (RAW;NEF....)
21 : Photoshop PSD (PSD)
22 : Vectorial objects (IEV)
23 : Layers (LYR)
24 : Layers and vectorial objects (ALL)
25 : Microsoft HD Photo (HDP;WDP)
26 : Video for Windows (AVI)
27 : Mpeg (MPG;MPEG)
28 : Windows Media Video (WMV)

ExtendedFilters specifies additional file formats (example: 'Fun Bitmap|*.fun;*.fan'). Title specifies the dialog title. Empty string means operating system default value. Filter specifies file extensions to enable (i.e. 'JPEG Bitmap (JPG)|*.jpg|CompuServe Bitmap (GIF)|*.gif').

ExecuteOpenDialog returns null string (") if the user clicks on Cancel.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
function ExecuteSaveDialog(const InitialDir:
WideString; const InitialFileName: WideString;
AlwaysAnimate: Boolean; FilterIndex: integer;
const ExtendedFilters: WideString; const Title:
WideString; const Filter: WideString =
''):WideString;
```

DESCRIPTION

The ExecuteSaveDialog shows and executes the save dialog. It encapsulates the TSaveImageEnDialog component. InitialDir is the starting directory. InitialFileName is the default file name with extension. AlwaysAnimate if true auto-animate GIF and AVI FilterIndex specifies what file format to select by default. The InitialFileName extension, if specified, in the second parameter, will over-ride the specified index. Valid values are:

1 : TIFF Bitmap (TIF;TIFF;FAX)
2 : GIF (GIF) - if enabled
3 : JPEG Bitmap (JPG;JPEG;JPE)
4 : PaintBrush (PCX)
5 : Windows Bitmap (BMP;DIB;RLE)
6 : Windows Icon (ICO)
7 : Portable Network Graphics (PNG)
8 : Targa Bitmap (TGA;TARGA;VDA;ICB;VST;PIX)
9 : Portable Pixmap, GreyMap, BitMap (PXM;PPM;PGM;PBM)
10 : Wireless Bitmap (WBMP)
11 : Jpeg2000 (JP2)
12 : Jpeg2000 Code Stream (J2K;JPC;J2C)
13 : PostScript Level 2 (PS;EPS)

14 : Adobe PDF (PDF)
15 : Multipage PCX (DCX)
16 : Photoshop PSD (PSD)
17 : Vectorial objects (IEV)
18 : Layers (LYR)
19 : Layers and objects (ALL)
20 : Microsoft HD Photo
21 : Video for Windows (AVI)

ExtendedFilters specifies additional file formats (example: 'Fun Bitmap|*.fun;*.fan'). Title specifies the dialog title. Empty string means operating system default value. Filter specifies file extensions to enable (i.e. 'JPEG Bitmap (JPG)|*.jpg|CompuServe Bitmap (GIF)|*.gif').

ExecuteSaveDialog returns a null string (") if the user clicks on Cancel.

DECLARATION

```
property MsgLanguage: TMsgLanguage;
```

DESCRIPTION

This property sets the language for Input/output previews (DoPreviews) and print dialogs (DoPrintPreviewDialog).

Unit ImageEnIO
TImageEnIO

DECLARATION

```
property PreviewFont: TFont;
```

DESCRIPTION

The PreviewFont property contains the font used in the DoPreviews dialog. Make sure the size of the font match label's length.

DECLARATION

```
property PreviewsParams: TIOPreviewsParams;
```

DESCRIPTION

This property specifies the features that the input/output preview dialog will have. If the SimplifiedParamsDialogs property is True (the default), the 'Advanced' button of open/save dialogs will show a simplified set of parameters. Warning: the default is True, to allow old style "advanced" dialogs set it to False.

DECLARATION

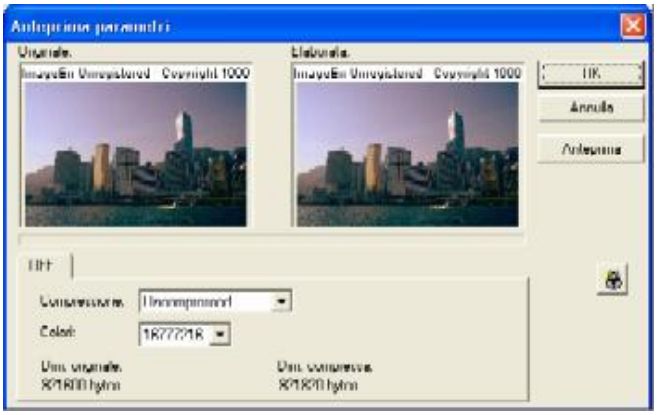
```
property SimplifiedParamsDialogs: boolean;
```

DESCRIPTION

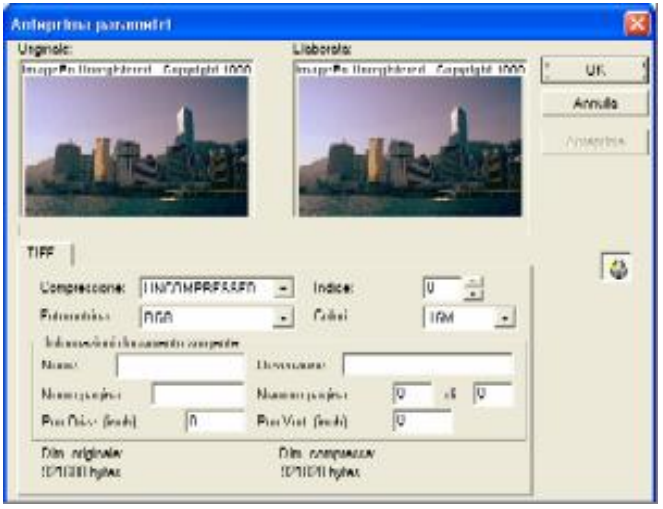
If the SimplifiedParamsDialogs property is True (the default), the 'Advanced' button of open/save dialogs will show a simplified set of parameters. Warning: the default is True, to allow old style "advanced" dialogs set it to False.

Example

This shows the 'Advanced' dialog for TIFF, using SimplifiedParamsDialogs=True:



This show the 'Advanced' dialog for TIFF, using SimplifiedParamsDialogs=False:



Printing

Declaration

```
function DoPrintPreviewDialog(DialogType:
TIEDialogType; const TaskName: WideString;
PrintAnnotations: Boolean; const Caption:
WideString): Boolean;
```

Description

DoPrintPreviewDialog shows the print preview dialog of type DialogType.

TaskName determines the text that appears listed in the Print Manager and on network header pages. If PrintAnnotation is true and the image contains Imaging Annotations they will be printed out. Caption specifies the caption of preview dialog.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure PreviewPrintImage(DestBitmap: TBitmap;
MaxBitmapWidth, MaxBitmapHeight: integer;
Printer: TPrinter; MarginLeft, MarginTop,
MarginRight, MarginBottom: double; VerticalPos:
TIEVerticalPos; HorizontalPos: TIEHorizontalPos;
Size: TIESize; SpecWidth, SpecHeight: double;
GammaCorrection: double);
```

DESCRIPTION

PreviewPrintImage draws the current image print preview, specifying margins, vertical position, horizontal position and size. It also paints a rectangle over the image that specifies the margin limits.

Parameter	Description
DestBitmap	Destination bitmap where to paint the preview. Resizes DestBitmap as needed.
MaxBitmapWidth	Maximum destination bitmap width. Preview will be stretched to match this size.
MaxBitmapHeight	Maximum destination bitmap height. Preview will be stretched to match this size.
Printer	This is the Printer object. PreviewPrintImage need it to know printer settings as orientation or page sizes.
MarginLeft	Left page margin in inches. Specifying zero causes no margin to be used.

Parameter	Description
MarginTop	Top page margin in inches. Specifying zero causes no margin to be used.
MarginRight	Right page margin in inches. Specifying zero causes no margin to be used.
MarginBottom	Bottom page margin in inches. Specifying zero causes no margin to be used.
VerticalPos	Determines how the image vertically aligns within the page.
HorizontalPos	Determines how the image horizontally aligns within the page.
Size	Determines the size of the image.
SpecWidth	Specifies the absolute width of the image in inches. Valid only when Size=iesSPECIFIEDSIZE.
SpecHeight	Specifies the absolute height of the image in inches. Valid only when Size=iesSPECIFIEDSIZE.
GammaCorrection	The gamma correction value, use 1.0 to disable gamma correction.

ImageEnIO

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure PrintImagePos(PrtCanvas: TCanvas; x,
y:double; Width, Height: double; GammaCorrection:
double);
```

DESCRIPTION

PrintImagePos prints the current image, specifying absolute position and size.

PrtCanvas is the printing canvas. The application can set this parameter from Printer.Canvas.

x, y is the top-left starting point, in inches.
Width, Height is the image size, in inches.

GammaCorrection is the gamma correction value. Use a value of 1.0 to disable gamma correction.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure PrintImage(PrtCanvas: TCanvas;
MarginLeft, MarginTop, MarginRight, MarginBottom:
double; VerticalPos: TIEVerticalPos;
HorizontalPos: TIEHorizontalPos; Size: TIESize;
SpecWidth, SpecHeight: double; GammaCorrection:
double);
```

DESCRIPTION

PrintImage prints the current image by specifying margins, vertical position, horizontal position and size.

Parameter	Description
PrtCanvas	This is the printing canvas. The application can set this parameter from Printer.Canvas.
MarginLeft	Left page margin in inches. Specifying zero causes no margin to be used.
MarginTop	Top page margin in inches. Specifying zero causes no margin to be used.
MarginRight	Right page margin in inches. Specifying zero causes no margin to be used.
MarginBottom	Bottom page margin in inches. Specifying zero causes no margin to be used.
VerticalPos	Determines how the image vertically aligns within the page.

Parameter	Description
HorizontalPos	Determines how the image horizontally aligns within the page.
Size	Determines the size of the image.
SpecWidth	Specifies the absolute width of the image in inches. Valid only when Size=iesSPECIFIEDSIZE.
SpecHeight	Specifies the absolute height of the image in inches. Valid only when Size=iesSPECIFIEDSIZE.
GammaCorrection	The gamma correction value, use 1.0 to disable gamma correction.

DECLARATION

```
property PrintingFilterOnSubsampling:
TResampleFilter;
```

DESCRIPTION

Specifies a filter when the image needs to be printed and it must be resampled. Filtering enhances the image quality but slows processing.

DECLARATION

```
property PrintPreviewParams:
TIOPrintPreviewParams;
```

DESCRIPTION

This property allows you to set/get parameters of print preview dialog. All measure units are specified by DialogsMeasureUnit property.

JPEG

DECLARATION

```
function InjectJpegEXIFStream(InputStream,
OutputStream: TStream): Boolean;
```

DESCRIPTION

InjectJpegEXIFStream replaces EXIF information of the InputStream stream (jpeg) with EXIF current in memory without loading or modifying the original image. OutputStream contains the modified stream.
The method returns false if it fails.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
function InjectJpegEXIF(const FileName:
WideString):Boolean;
```

DESCRIPTION

InjectJpegEXIF replaces EXIF information of the FileName Jpeg file with EXIF current in memory without loading or modifying the original image. The method returns false if it fails.

DECLARATION

```
function InjectJpegIPTCStream(InputStream,
OutputStream: TStream):Boolean;
```

DESCRIPTION

InjectJpegIPTCStream replaces IPTC information of the InputStream stream (Jpeg) with IPTC currently in memory without loading or modifying the original image. OutputStream contains the modified stream. The method returns false if it fails.

Declaration

```
function InjectJpegIPTC(const FileName:
WideString):Boolean;
```

Description

InjectJpegIPTC replaces IPTC information of the FileName jpeg file with IPTC current in memory without loading or modifying the original image. The method returns False on fail.

Declaration

```
procedure LoadFromFileJpeg(const FileName:
WideString);
```

Description

LoadFromFileJpeg loads the image from a JPEG file. If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension.

Unit ImageEnIO
TImageEnIO

DECLARATION

```
procedure LoadFromStreamJpeg(Stream: TStream);
```

DESCRIPTION

LoadFromStreamJpeg loads the image from a JPEG stream. If the stream does not represent a valid image format, the Aborting property will be true. If StreamHeaders property is true, the stream must have a special header (saved with SaveToStreamJPEG).

DECLARATION

```
procedure SaveToFileJpeg(const FileName:
WideString);
```

DESCRIPTION

SaveToFileJpeg saves the current image to a file in JPEG format. FileName is the file name, with extension.

Unit ImageEnIO
TImageEnIO

DECLARATION

```
procedure SaveToStreamJpeg(Stream: TStream);
```

DESCRIPTION

SaveToStreamJpeg saves the current image as JPEG in the Stream. If StreamHeaders property is true, it adds an additional special header as needed for multi-image streams.

JPEG 2000

Declaration

```
procedure LoadFromFileJ2K(const FileName:
WideString);
```

Description

LoadFromFileJ2K loads an image from a Jpeg2000 (J2K code stream) file. If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension.

Declaration

```
procedure LoadFromFileJP2(const FileName:
WideString);
```

Description

LoadFromFileJP2 loads an image from a Jpeg2000 (JP2) file. If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure LoadFromStreamJ2K(Stream: TStream);
```

DESCRIPTION

LoadFromStreamJ2K loads an image from a Jpeg2000 (J2K code stream) stream. If the file does not represent a valid image format, the Aborting property will be true.

DECLARATION

```
procedure LoadFromStreamJP2(Stream: TStream);
```

DESCRIPTION

LoadFromStreamJP2 loads an image from a Jpeg2000 (JP2) stream. If the file does not represent a valid image format, the Aborting property will be true.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure SaveToFileJ2K(const FileName:
WideString);
```

DESCRIPTION

SaveToFileJ2K saves the current image in Jpeg2000 (J2K code stream) format to the FileName file. FileName is the file name, with extension.

DECLARATION

```
procedure SaveToFileJP2(const FileName:
WideString);
```

DESCRIPTION

SaveToFileJP2 saves the current image in Jpeg2000 (JP2) format to the FileName file. FileName is the file name, with extension.

DECLARATION

```
procedure SaveToStreamJ2K(Stream: TStream);
```

DESCRIPTION

SaveToStreamJ2K saves the current image as Jpeg2000 (J2K code stream) in the Stream.

DECLARATION

```
procedure SaveToStreamJP2(Stream: TStream);
```

DESCRIPTION

SaveToStreamJP2 saves the current image as Jpeg2000 (JP2) in the Stream.

GIF

DECLARATION

```
function InsertToFileGif(const FileName:
WideString): integer;
```

DESCRIPTION

This function inserts a frame in a GIF file and makes it animated. The file must exist. FileName is the file name with extension. InsertToFileGif returns the number of images inside the specified file.

DECLARATION

```
function LoadFromFileGIF(const FileName:
WideString): integer;
```

DESCRIPTION

LoadFromFileGIF loads an image from a GIF file (87a, 89a and animated GIF).

If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension. Returns number of images contained in GIF file (number of frames if this is an animated GIFF).

Unit ImageEnIO

TImageEnIO

DECLARATION

```
function LoadFromStreamGIF(Stream: TStream): integer;
```

DESCRIPTION

LoadFromStreamGIF loads an image from a stream.

If the stream does not represent a valid image format, the Aborting property will be true. Returns number of images contained in the stream (number of frames if this is an animated stream). If StreamHeaders property is true, the stream must have a special header (saved using SaveToStreamGIF).

DECLARATION

```
procedure SaveToFileGif(const FileName: WideString);
```

DESCRIPTION

SaveToFileGif saves the image in a GIF file (89a). The GIF will have a single image and it won't be marked as animated. FileName is the file name with extension.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure SaveToStreamGif(Stream: TStream);
```

DESCRIPTION

SaveToStreamGif saves current image as GIF in the Stream.

If StreamHeaders property is True, it adds an additional special header as needed for multi-image streams.

DECLARATION

```
function ReplaceFileGIF(const FileName:
WideString):integer;
```

DESCRIPTION

This function saves the current image in the specified multi-page GIF file, replacing only the image at the index specified with Params.GIF_ImageIndex.

Adobe PSD

Declaration

```
procedure LoadFromFilePSD(const FileName: WideString);
```

Description

LoadFromFilePSD loads an image from a Adobe PSD file. If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension.

PSD can contain multiple layers. If you want load separated layers write:

```
ImageEnView1.IO.Params.PSD_LoadLayers:=true;
ImageEnView1.IO.LoadFromFilePSD(filename);
```

If you want load only the merged image (default) write:

```
ImageEnView1.IO.Params.PSD_LoadLayers:=false;
ImageEnView1.IO.LoadFromFilePSD(filename);
```

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure LoadFromStreamPSD(Stream: TStream);
```

DESCRIPTION

LoadFromStreamPSD loads an image from a Adobe PSD stream. If the file does not represent a valid image format, the Aborting property will be true.

PSD can contain multiple layers. If you want load separated layers write:

```
ImageEnView1.IO.Params.PSD_LoadLayers:=true;
ImageEnView1.IO.LoadFromStreamPSD(stream);
```

If you want load only the merged image (default) write:

```
ImageEnView1.IO.Params.PSD_LoadLayers:=false;
ImageEnView1.IO.LoadFromStreamPSD(stream);
```

DECLARATION

```
procedure SaveToFilePSD(const FileName:
WideString);
```

DESCRIPTION

SaveToFilePSD saves current image or all layers as PSD file format. PSD supports multiple layers, so it can store position and other useful info (like layer name). PSD also have a merged representation of the image to fast preview the merged result.

DECLARATION

```
procedure SaveToStreamPSD(Stream: TStream);
```

DESCRIPTION

SaveToStreamPSD saves current image or all layers as PSD file format. PSD supports multiple layers, so it can store position and other useful info (like layer name). PSD also have a merged representation of the image to fast preview the merged result.

HDP - Microsoft HD Photo

DECLARATION

```
procedure LoadFromFileHDP(const FileName:
WideString);
```

DESCRIPTION

LoadFromFileHDP loads an image from Microsoft HD Photo file.

If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension. Requires: Windows XP (SP2) with .Net 3.0, Windows Vista

Unit ImageEnIO
TImageEnIO

DECLARATION

```
procedure LoadFromStreamHDP(Stream: TStream);
```

DESCRIPTION

LoadFromStreamPSD loads an image from a Microsoft HD Photo stream. If the file does not represent a valid image format, the Aborting property will be true. LoadFromStreamHDP requires Windows XP (SP2) with .Net 3.0, Windows Vista.

DECLARATION

```
procedure SaveToFileHDP(const FileName:
WideString);
```

DESCRIPTION

SaveToFileHDP saves current image or all layers as Microsoft HD Photo file format. SaveToFileHDP requires Windows XP (SP2) with .Net 3.0, Windows Vista.

DECLARATION

```
procedure SaveToStreamHDP(Stream: TStream);
```

DESCRIPTION

SaveToStreamHDP saves current image or all layers as Microsoft HD Photo file format. SaveToStreamHDP requires: Windows XP (SP2) with .Net 3.0, Windows Vista.

TIFF

Declaration

```
function InjectTIFFEXIF(const FileName:
WideString; pageIndex: integer): Boolean;
function InjectTIFFEXIF(const InputFileName,
OutputFileName: WideString; pageIndex: integer):
Boolean;
function InjectTIFFEXIF(InputStream,
OutputStream: TStream; pageIndex: integer):
Boolean;
```

Description

InjectTIFFEXIF replaces EXIF meta-tags of InputStream or FileName with EXIF currently in memory without loading or modifying the original image. InjectTIFFEXIF works only with TIFF and Microsoft HDPhoto file formats. The method returns false if it fails

DECLARATION

```
function InsertToFileTIFF(const FileName:
WideString): integer;
```

DESCRIPTION

Insert a frame in a TIFF file at position specified by <TIOParamsVals.ImageIndex>. The file must exist. FileName is the file name with extension. InsertToFileTIFF returns the number of images inside the file.

Example

```
// Create a Multi-Page file
ImageEnView1.IO.LoadFromFile('page1.tif');
ImageEnView1.IO.Params.TIFF_ImageIndex:=0;
// increment this for each page
ImageEnView1.IO.InsertToFileTIFF('multipage.tif');
```

DECLARATION

```
function InsertToStreamTIFF(Stream: TStream):
integer;
```

DESCRIPTION

Insert a frame in a TIFF stream at position specified by <TIOParamsVals.ImageIndex>.. The stream position must be at the beginning.

InsertToStreamTIFF returns the number of images inside the file.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
function LoadFromFileTIFF(const FileName:
WideString): integer;
```

DESCRIPTION

LoadFromFileTIFF loads an image from a TIFF file (rev. 6.0, Packbits, LZW, CCITT G.3 and G.4). If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension. Returns number of images contained in TIFF file.

DECLARATION

```
function LoadFromStreamTIFF(Stream: TStream):
integer;
```

DESCRIPTION

LoadFromStreamTIFF loads the image from a TIFF stream (rev. 6.0, Packbits, LZW, CCITT G.3 and G.4).

If the stream does not represent a valid image format, the Aborting property will be true. Returns number of images contained in the stream. If StreamHeaders property is true, the stream must have a special header (saved using SaveToStreamTIFF).

Declaration

```
function ReplaceFileTIFF(const FileName: WideString): integer;
```

Description

This function saves current image replacing to the specified multi-page TIFF file, replacing only the image at index specified with Params.TIFF_ImageIndex.

Declaration

```
function ReplaceStreamTIFF(Stream: TStream): integer;
```

Description

This function saves current image replacing to the specified multi-page TIFF stream, replacing only the image at index specified with Params.TIFF_ImageIndex. Stream position must be at the beginning of the TIFF content.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure SaveToFileTIFF(const FileName:
WideString);
```

DESCRIPTION

SaveToFileTIFF saves the current image in TIFF format to a file. FileName is the file name, with extension.

DECLARATION

```
procedure SaveToStreamTIFF(Stream: TStream);
```

DESCRIPTION

SaveToStreamTIFF saves the current image with PCX format to the specified Stream. If StreamHeaders property is true, it adds an additional special header as needed for multi-image streams.

BMP

DECLARATION

```
procedure LoadFromFileBMP(const FileName:
WideString);
```

DESCRIPTION

LoadFromFileBMP loads an image from a BMP file. If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension.

DECLARATION

```
procedure LoadFromStreamBMP(Stream: TStream);
```

DESCRIPTION

LoadFromStreamBMP loads the image from a BMP stream.

If the stream does not represent a valid image format, the Aborting property will be true. If StreamHeaders property is true, the stream must have a special header (saved with SaveToStreamBMP).

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure SaveToFileBMP(const FileName:
WideString);
```

DESCRIPTION

SaveToFileBMP saves the current image in BMP format to a file. FileName is the file name with extension.

DECLARATION

```
procedure SaveToStreamBMP(Stream: TStream);
```

DESCRIPTION

SaveToStreamBMP saves current image as BMP in the Stream. If StreamHeaders property is true, it adds an additional special header as needed for multi-image streams.

PNG

DECLARATION

```
procedure LoadFromFilePNG(const FileName:
WideString);
```

DESCRIPTION

LoadFromFilePNG loads an image from a PNG file. If the file does not represent a valid image format, the Aborting property will be True. FileName is the file name with extension.

DECLARATION

```
procedure LoadFromStreamPNG(Stream: TStream);
```

DESCRIPTION

LoadFromStreamPNG loads the image from a PNG stream. If the stream does not represent a valid image format, the Aborting property is True.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure SaveToFilePNG(const FileName:
WideString);
```

DESCRIPTION

SaveToFilePNG saves the current image to a file in PNG format.
FileName is the file name, with extension.

Example

```
// Set best compression
ImageEnView1.IO.Params.PNG_Filter:=ioPNG_FILTER_P
AETH;
ImageEnView1.IO.Params.PNG_Compression:=9;
// Save PNG
ImageEnView1.IO.SaveToFilePNG('max.png');
```

ImageEnIO

Unit ImageEnIO
TImageEnIO

DECLARATION

```
procedure SaveToStreamPNG(Stream: TStream);
```

DESCRIPTION

SaveToStreamPNG saves the current image as PNG in the Stream.

Example

```
// Set best compression
ImageEnView1.IO.Params.PNG_Filter:=ioPNG_FILTER_P
AETH;
ImageEnView1.IO.Params.PNG_Compression:=9;
// Save PNG
ImageEnView1.IO.SaveToStreamPNG(Stream);
```

DCX

DECLARATION

```
procedure LoadFromFileDCX(const FileName:
WideString);
```

DESCRIPTION

LoadFromFileDCX loads a DCX file. FileName is the file name. You can set the page to load using the Params.DCX_ImageIndex property.

DECLARATION

```
procedure LoadFromStreamDCX(Stream: TStream);
```

DESCRIPTION

LoadFromStreamDCX loads a DCX file from specified Stream. You can set the page to load using the Params.DCX_ImageIndex property.

DECLARATION

```
procedure SaveToStreamDCX(Stream: TStream);
```

DESCRIPTION

SaveToStreamDCX saves current image in DCX format to the specified stream.

DECLARATION

```
procedure SaveToFileDCX(const FileName:
WideString);
```

DESCRIPTION

SaveToFileDCX saves current image in DCX format to the specified file.

DECLARATION

```
procedure InsertToFileDCX(const FileName:
WideString);
```

DESCRIPTION

Inserts current image in DCX format to the specified file, to the position specified by Params.DCX_ImageIndex.

CUR

DECLARATION

```
procedure LoadFromFileCUR(const FileName:
WideString);
```

DESCRIPTION

LoadFromFileCUR loads an image from a CUR file.
If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension.

DECLARATION

```
procedure LoadFromStreamCUR(Stream: TStream);
```

DESCRIPTION

LoadFromStreamCUR loads a CUR image from a stream.
If the stream does not represent a valid image format, the Aborting property is true.

ICO

DECLARATION

```
procedure LoadFromFileICO(const FileName:
WideString);
```

DESCRIPTION

LoadFromFileICO loads an image from an ICO file.

If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension.

DECLARATION

```
procedure LoadFromStreamICO(Stream: TStream);
```

DESCRIPTION

LoadFromStreamICO loads an ICO image from a stream.
If the stream does not represent a valid image format, the Aborting property will be true.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure SaveToFileICO(const FileName:
WideString);
```

DESCRIPTION

SaveToFileICO saves the image to a file. ICO format allows storing multiple images with several resolutions and sizes. ImageEn creates for you the sub-images to save, you have only to set the Params.ICO_BitCount and Params.ICO_Sizes arrays.

If you want control each image separately, use IEWriteICOImages function.

DECLARATION

```
procedure SaveToStreamICO(Stream: TStream);
```

DESCRIPTION

SaveToStreamICO saves the image to an ICO stream. ICO format allows storing multiple images with several resolutions and sizes. ImageEn creates for you the sub-images to save; you have only to set the Params.ICO_BitCount and Params.ICO_Sizes arrays.

If you want control each image separately use IEWriteICOImages function.

PCX

DECLARATION

```
procedure LoadFromFilePCX(const FileName:
WideString);
```

DESCRIPTION

LoadFromFilePCX loads the image from a PCX file. If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension.

DECLARATION

```
procedure LoadFromStreamPCX(Stream: TStream);
```

DESCRIPTION

LoadFromStreamPCX loads the image from a PCX stream. If the stream does not represent a valid image format, the Aborting property will be true. If StreamHeaders property is true, the stream must have a special header (saved using SaveToStreamPCX).

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure SaveToFilePCX(const FileName:
WideString);
```

DESCRIPTION

SaveToFilePCX saves the current image in PCX format to a file. FileName is the file name, with extension.

DECLARATION

```
procedure SaveToStreamPCX(Stream: TStream);
```

DESCRIPTION

SaveToStreamPCX saves the current image as PCX in the Stream. If StreamHeaders property is true, it adds an additional special header as needed for multi-image streams.

Unit ImageEnIO

TImageEnIO

WMF, EMF

DECLARATION

```
procedure ImportMetafile(const FileName:
WideString; Width, Height: integer; WithAlpha:
Boolean);
procedure ImportMetafile(Stream: TStream; Width:
integer=-1; Height: integer=-1; WithAlpha:
Boolean=true);
procedure ImportMetaFile(meta: TMetaFile; Width,
Height: integer; WithAlpha: Boolean);
```

DESCRIPTION

ImportMetafile imports a WMF or EMF vectorial image.

Width and Height are required image sizes (the rectangle where the vectorial image will be painted). To maintain the image aspect ratio set only one size, assigning -1 to the other (ex. ImportMetafile('axi.emf',-1,500)).

If WithAlpha is true then ImportMetaFile creates an alpha channel, making visible only the metafile content. If the file does not represent a valid image format, ImportMetafile raises an EInvalidGraphic exception.

Important: ImportMetafile converts vectorial image to raster image, so we have to limit the rasterized image sizes.
The maximum allowed size is stored in the public field named iegMaxImageEMFSize.

DECLARATION

```
procedure MergeMetafile(const FileName:
WideString; x, y, Width, Height: integer);
```

DESCRIPTION

MergeMetafile loads a metafile and draws it at the x, y position using Width and Height sizes. If Width or Height is -1 this one is autocalculated.

TGA

DECLARATION

```
procedure LoadFromFileTGA(const FileName:
WideString);
```

DESCRIPTION

LoadFromFileTGA loads an image from a TGA file. If the file does not represent a valid image format, the Aborting property will be true.

FileName is the file name with extension.

DECLARATION

```
procedure LoadFromStreamTGA(Stream: TStream);
```

DESCRIPTION

LoadFromStreamTGA loads an image from a TGA stream. If the stream does not represent a valid image format, the Aborting property will be true. Stream is the image stream.

DECLARATION

```
procedure SaveToFileTGA(const FileName:
WideString);
```

DESCRIPTION

SaveToFileTGA saves the current image in TGA format to a file. FileName is the file name, with extension.

DECLARATION

```
procedure SaveToStreamTGA(Stream: TStream);
```

DESCRIPTION

SaveToStreamTGA saves the current image in TGA format to stream. Stream is the image stream.

PXM (PPM, PBM,PGM)

DECLARATION

```
procedure LoadFromFilePXM(const FileName:
WideString);
```

DESCRIPTION

LoadFromFilePXM and LoadFromStreamPXM methods load an image from a PBM, PGM, PPM file or stream. If the file/stream does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension.

DECLARATION

```
procedure LoadFromStreamPXM(Stream: TStream);
```

DESCRIPTION

LoadFromFilePXM and LoadFromStreamPXM methods load an image from a PBM, PGM, PPM file or stream. If the file/stream does not represent a valid image format, the Aborting property will be true. Stream is the source stream.

Unit ImageEnIO
TImageEnIO

DECLARATION

```
procedure SaveToFilePXM(const FileName:
WideString);
```

DESCRIPTION

SaveToFilePXM or SaveToStreamPXM saves the current image in PBM, PGM or PPM format to file or stream. FileName is the file name, with extension.

If the PBM (Portable Bitmap) contains only 1 bpp images (black/white), then the values must be Params.BitsPerPixel=1 and Params.SamplesPerPixel=1.

If the PGM (Portable Graymap) contains only 8 bpp images (gray scale), then the values must be Params.BitsPerPixel=8 and Params.SamplesPerPixel=1. If the PPM (Portable Pixmap) contains only 24 bpp images (true color), then the values must be Params.BitsPerPixel=8 and Params.SamplesPerPixel=3.

DECLARATION

```
procedure SaveToStreamPXM(Stream: TStream);
```

DESCRIPTION

SaveToFilePXM or SaveToStreamPXM saves the current image in PBM, PGM or PPM format to file or stream. Stream is the source stream.

If the PBM (Portable Bitmap) contains only 1 bpp images (black/white), then the values must be Params.BitsPerPixel = 1 and Params.SamplesPerPixel = 1. If the PGM (Portable Graymap) contains only 8 bpp images (gray scale), then the values must be Params.BitsPerPixel=8 and Params.SamplesPerPixel = 1. If the PPM (Portable Pixmap) contains only 24 bpp images (true color), then the values must be Params.BitsPerPixel=8 and Params.SamplesPerPixel = 3.

AVI

DECLARATION

```
procedure CloseAVIFile;
```

DESCRIPTION

CloseAVIFile closes the AVI file opened with OpenAVIFile or CreateAVIFile.

Unit ImageEnIO
TImageEnIO

DECLARATION

```
function CreateAVIFile(const FileName:
WideString; rate: integer; const codec:
AnsiString): TIECreateAVIFileResult;
```

DESCRIPTION

CreateAVIFile creates a new AVI file using FileName file name. The rate parameter specifies the number of frames per second. Codec specifies the compression codec to use (must be installed on system) as four characters length string.

For example:
'cvid' : cinepak by Radius
'msvc' : Microsoft Video 1
'mp42' : Microsoft MPEG4 V2
more codecs at http://www.fourcc.org or http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwmt/html/registeredfourcccodesandwaveformats.asp (or at msdn.microsoft.com searching for registered fourcc codes and wave formats).

You can save each frame to the created AVI file using the SaveToAVI method. Finally, call CloseAVIFile to close the file.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
function IsOpenAVI: Boolean;
```

DESCRIPTION

Returns True when OpenAVIFile has currently open a file.

DECLARATION

```
procedure LoadFromAVI(FrameIndex: integer);
```

DESCRIPTION

LoadFromAVI read the FrameIndex frame opened with the OpenAVIFile method.

DECLARATION

```
function OpenAVIFile(const FileName:
WideString):integer;
```

DESCRIPTION

OpenAVIFile opens an AVI file, without reading frames. OpenAVIFile returns the frame count contained in the AVI. See also CloseAVIFile.

DECLARATION

```
procedure SaveToAVI;
```

DESCRIPTION

SaveToAVI saves the image to the current open AVI file (open using CreateAVIFile). We suggest setting Params.BitsPerSample and Params.SamplesPerPixel before calling SaveToAVI, because AVI requires all images have same color depth.

WBMP

DECLARATION

```
procedure LoadFromFileWBMP(const FileName:
WideString);
```

DESCRIPTION

LoadFromFileWBMP loads an image from a WBMP (Wireless bitmap) file. If the file does not represent a valid image format, the Aborting property will be true. FileName is the file name with extension.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure LoadFromStreamWBMP(Stream: TStream);
```

DESCRIPTION

LoadFromStreamWBMP loads an image from a WBMP (Wireless bitmap) stream. If the file does not represent a valid image format, the Aborting property will be true.

DECLARATION

```
procedure SaveToFileWBMP(const FileName:
WideString);
```

DESCRIPTION

SaveToFileWBMP saves the current image in WBMP (Wireless bitmap) format to file. FileName is the file name, with extension.

DECLARATION

```
procedure SaveToStreamWBMP(Stream: TStream);
```

DESCRIPTION

SaveToStreamWBMP saves the current image as WBMP (Wireless bitmap) in the Stream.

PostScript (PS)

DECLARATION

```
procedure ClosePSFile;
```

DESCRIPTION

ClosePSFile closes the currently open PostScript file. You can create a PostScript file using CreatePSFile then add pages using SaveToPS and finally close the file using ClosePSFile. The resulting PS file will contain only images, aligned to the upper-left side of the paper.

DECLARATION

```
procedure CreatePSFile(const FileName:
WideString);
```

DESCRIPTION

CreatePSFile creates a new PostScript file using FileName path and file name. You can add pages using SaveToPS and finally close the file using ClosePSFile. The PS file will contain only images, aligned to the upper-left side of the paper.

DECLARATION

```
procedure SaveToFilePS(const FileName:
WideString);
```

DESCRIPTION

SaveToFilePS saves the current image to FileName (filename). The resulting PostScript file will have only one page.

DECLARATION

```
procedure SaveToPS;
```

DESCRIPTION

SaveToPS saves the current image to an open PostScript file. You can create a PostScript file using CreatePSFile then add pages using SaveToPS and finally close the file using ClosePSFile. The resulting PS file will contain only images, aligned to the upper-left side of the paper.

DECLARATION

```
procedure SaveToStreamPS(Stream: TStream);
```

DESCRIPTION

SaveToStreamPS saves the current image to Stream. The resulting PostScript file will have only one page.

Unit ImageEnIO

TImageEnIO

Adobe PDF

DECLARATION

```
procedure ClosePDFFile;
```

DESCRIPTION

ClosePDFFile closes the current open Adobe PDF file. You can create a PDF file using CreatePDFFile then add pages using SaveToPDF and finally close the file using ClosePDFFile. The resulting PDF file will contain only images, aligned to the upper-left side of the paper.

DECLARATION

```
procedure CreatePDFFile(const FileName:
WideString);
```

DESCRIPTION

CreatePDFFile creates a new Adobe PDF file using FileName path and file name. You can add pages using SaveToPDF and finally close the file using ClosePDFFile. The resulting PDF file will contain only images, aligned to the upper-left side of the paper.

ImageEnIO

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure SaveToFilePDF(const FileName:
WideString);
```

DESCRIPTION

SaveToFilePDF saves current image to FileName (filename). The resulting Adobe PDF file will have only one page.

DECLARATION

```
procedure SaveToPDF;
```

DESCRIPTION

SaveToPDF saves the current image to an open Adobe PDF file. You can create a PDF file using CreatePDFFile then add pages using SaveToPDF and finally close the file using ClosePDFFile. The resulting PDF file will contain only images, aligned to the upper-left side of the paper.

DECLARATION

```
procedure SaveToStreamPDF(Stream: TStream);
```

DESCRIPTION

SaveToStreamPDF saves the current image to Stream. The resulting Adobe PDF file will have only one page.

Unit ImageEnIO

TImageEnIO

RAW Camera

Declaration

```
procedure LoadFromFileRAW(const FileName:
WideString);
```

Description

LoadFromFileRAW loads an image from a file as Camera RAW. See also List of supported Camera RAW formats.

Declaration

```
procedure LoadFromStreamRAW(Stream: TStream);
```

Description

LoadFromStreamRAW loads an image from a stream as Camera RAW. See also List of supported Camera RAW formats

Declaration

```
procedure LoadJpegFromFileCRW(const FileName:
WideString);
```

Description

Extract and load the large Jpeg encapsulated inside a CRW (canon RAW) file.

ImageEnIO

MediaFiles (AVI, MPEG, WMV...)

DECLARATION

```
procedure CloseMediaFile;
```

DESCRIPTION

CloseMediaFile closes a video file open with OpenMediaFile.

DECLARATION

```
function IsOpenMediaFile: Boolean;
```

DESCRIPTION

Returns true when OpenMediaFile has currently open a file.
Procedure LoadFromMediaFile(FrameIndex: integer);

DECLARATION

```
procedure LoadFromMediaFile(FrameIndex: integer);
```

DESCRIPTION

LoadFromMediaFile loads the specified frame from a media file open using OpenMediaFile. The first frame has index 0.

DECLARATION

```
function OpenMediaFile(const FileName:
WideString):integer;
```

DESCRIPTION

OpenMediaFile opens a video file using DirectShow which allows you to load AVI, Mpeg, WMV and other DirectX supported formats. It opens the file, but doesn't actually get the image. To get a frame use the LoadFromMediaFile method. To close the media file use CloseMediaFile.

Real RAW (not Camera RAW)

DECLARATION

```
procedure LoadFromFileBMPRAW(const FileName:
WideString);
```

DESCRIPTION

LoadFromFileBMPRAW loads a file as RAW. This is not camera RAW, but real RAW (named in ImageEn as BMPRAW) where you can specify the channels order, placement, alignment, size, etc.

Unit ImageEnIO

TImageEnIO

DECLARATION

```
procedure LoadFromStreamBMPRAW(Stream: TStream);
```

DESCRIPTION

LoadFromStreamBMPRAW loads an image from stream as RAW. This is not camera RAW, but real RAW (named in ImageEn as BMPRAW) where you can specify the channels order, placement, alignment, size, etc.

Demo
Inputoutput/RealRAW

DECLARATION

```
procedure SaveToFileBMPRAW(const FileName:
WideString);
```

DESCRIPTION

SaveToFileBMPRAW saves current image as RAW to file. This is not camera RAW, but real RAW (named in ImageEn as BMPRAW) where you can specify the channels order, placement, alignment, size, etc.

Demo
Inputoutput/RealRAW

Unit ImageEnIO

TImageEnIO

Declaration

```
procedure SaveToStreamBMPRAW(Stream: TStream);
```

Description

SaveToStreamBMPRAW saves current image as RAW to stream. This is not camera RAW, but real RAW (named in ImageEn as BMPRAW) where you can specify the channels order, placement, alignment, size, etc.

Demo
Inputoutput/RealRAW

DICOM Medical Imaging

Declaration

```
procedure LoadFromFileDICOM(const FileName:
WideString);
```

Description

LoadFromFileDICOM loads a DICOM image. This method is necessary when the DICOM file hasn't extension and hasn't a valid DICOM header, but you are sure that is a DICOM file. DICOM parameters are stored in DICOM_Tags.

DECLARATION

```
procedure LoadFromStreamDICOM(Stream: TStream);
```

DESCRIPTION

LoadFromStreamDICOM loads a DICOM image from stream. DICOM parameters are stored in DICOM_Tags.

Events

DECLARATION

```
property OnAcquireBitmap: TIEAcquireBitmapEvent;
```

DESCRIPTION

This event occurs whenever a new bitmap is acquired from multipage Twain acquisition.

DECLARATION

```
property OnAcquireClose: TNotifyEvent;
```

DESCRIPTION

This event occurs when the user closes the acquire dialog, open using TwainAcquireOpen.

DECLARATION

```
property OnDoPreviews: TIEDoPreviewsEvent;
```

DESCRIPTION

OnDoPreviews occurs just before the default dialog box (Advanced button in save dialog) shows. You can disable the dialog box by just setting Handled to true.

DECLARATION

```
property OnFinishWork: TNotifyEvent;
```

DESCRIPTION

OnFinishWork occurs when an input/output task ends. It is useful for resetting progress bars, or to know when a thread ends in a asynchronous mode.

DECLARATION

```
property OnIOPreview: TIEIOPreviewEvent;
```

DESCRIPTION

The OnIOPreview event is called before the preview form is displayed (running DoPreviews method).

Unit ImageEnIO

TImageEnIO

DECLARATION

```
property OnProgress: TIEProgressEvent;
```

DESCRIPTION

OnProgress event is called on input/output operations.

Chapter 7

TIOParamVals

TIOParamsVals object contains all parameters for all image file formats handled by ImageEn, such as compression type, text comments, bit per sample, samples per pixel, etc. You can set one or more of these parameters before saving or getting them after loading with the exception of “export only” file formats like PDF, PS, etc. which can be only saved: so you cannot load parameters from them.

See also

Params
IOParams

Unit ImageENIO
TIOParamVals

Properties and Methods

Properties Storage Handling

Assign	Create	LoadFromFile
LoadFromStream	ResetEXIF	ResetInfo
SaveToFile	SaveToStream	SetDefaultParams
UpdateEXIFThumbnail		

Common Parameters

BitsPerSample	ColorMapCount	ColorMap
DefaultICCProfile	Dpi	DpiX
DpiY	EnableAdjustOrientation	FileName
FileTypeStr	FileType	GetThumbnail
Height	ImageCount	ImageIndex
ImagingAnnot	InputICCProfile	IsResource
OutputICCProfile	OriginalHeight	OriginalWidth
SamplesPerPixel	Width	

BMP

BMP_Compression	BMP_HandleTransparency	BMP_Version

GIF

GIF_Action	GIF_Background	GIF_Comments
GIF_DelayTime	GIF_FlagTranspColor	GIF_ImageCount

GIF_ImageIndex GIF_Interlaced GIF_Ratio

GIF_TranspColor GIF_Version GIF_WinHeight

GIF_WinWidth GIF_XPos GIF_YPos

JPEG

JPEG_ColorSpace JPEG_DCTMethod JPEG_CromaSubsampling

JPEG_EnableAdjustOrientation JPEG_GetExifThumbnail JPEG_MarkerList

JPEG_OptimalHuffman JPEG_OriginalWidth JPEG_OriginalHeight

JPEG_Progressive JPEG_Quality JPEG_Scale_Used

JPEG_Scale JPEG_Smooth JPEG_WarningCode

JPEG_WarningTot

JPEG 2000

J2000_ColorSpace J2000_Rate J2000_ScalableBy

PCX

PCX_Compression PCX_Version

Adobe PSD

PSD_LoadLayers PSD_ReplaceLaye PSD_HasPremultipliedAlpha

HDP

HDP_ImageQuality HDP_Lossless

Unit ImageENIO
TIOParamVals

TIFF

TIFF_ByteOrder
TIFF_Compression
TIFF_DocumentName
TIFF_EnableAdjustOrientation
TIFF_GetTile
TIFF_FillOrder
TIFF_ImageCount
TIFF_ImageDescription
TIFF_ImageIndex
TIFF_JPEGColorSpace
TIFF_JPEGQuality
TIFF_Orientation
TIFF_PageCount
TIFF_PageName
TIFF_PageNumber
TIFF_PhotometInterpret
TIFF_PlanarConf
TIFF_SubIndex
TIFF_XPos
TIFF_YPos
TIFF_ZIPCompression

CUR

CUR_Background
CUR_ImageIndex
CUR_XHotSpot
CUR_YHotSpot

DCX

DCX_ImageIndex

ICO

ICO_Background
ICO_BitCount
ICO_ImageIndex
ICO_Sizes

Unit ImageENIO
TIOParamVals

PNG

PNG_Background	PNG_Compression	PNG_Filter
PNG_Interlaced	PNG_TextKeys	PNG_TextValues

TGA

TGA_AspectRatio	TGA_Author	TGA_Background
TGA_Compressed	TGA_Date	TGA_Descriptor
TGA_Gamma	TGA_GrayLevel	TGA_ImageName
TGA_XPos	TGA_YPos	

PXM

PXM_Comments

AVI

AVI_FrameCount	AVI_FrameDelayTime

IPTC

IPTC_Info

Unit ImageENIO
TIOParamVals

EXIF Tags

EXIF_ApertureValue EXIF_Artist EXIF_Bitmap

EXIF_BrightnessValue EXIF_ColorSpace EXIF_CompressedBitsPerPixel

EXIF_Contrast EXIF_Copyright EXIF_DateTimeDigitize

EXIF_DateTimeOriginal EXIF_DateTime EXIF_DigitalZoomRatio

EXIF_ExifImageHeight EXIF_ExifImageWidth EXIF_ExifVersion

EXIF_ExposureBiasValue EXIF_ExposureIndex EXIF_ExposureMode

EXIF_ExposureProgram EXIF_ExposureTime EXIF_FileSource

EXIF_FlashPixVersion EXIF_Flash EXIF_FNumber

EXIF_FocalLength EXIF_FocalLengthIn35mmFil EXIF_FocalPlaneResolution

EXIF_FocalPlaneXResolution EXIF_FocalPlaneYRes EXIF_GainControl

EXIF_HasEXIFData EXIF_ImageDescription EXIF_ImageUniqueID

EXIF_ISOSpeedRatings EXIF_LightSource EXIF_Make

EXIF_MakerNote EXIF_MaxApertureValue EXIF_MeteringMode

EXIF_Model EXIF_Orientation EXIF_PrimaryChromaticities

EXIF_ReferenceBlackWhite EXIF_RelatedSoundFile EXIF_ResolutionUnit

EXIF_Saturation EXIF_SceneCaptureType EXIF_SceneType

EXIF_SensingMethod EXIF_Sharpness EXIF_ShutterSpeedValue

EXIF_Software EXIF_SubjectDistanceRange EXIF_SubjectDistance

EXIF_SubsecTimeDigitized EXIF_SubsecTimeOriginal EXIF_SubsecTime

EXIF_UserCommentCode EXIF_UserComment EXIF_WhiteBalance

Unit ImageENIO
TIOParamVals

EXIF Tags

EXIF_WhitePoint EXIF_XResolution EXIF_YCbCrCoefficient

EXIF_YCbCrPositioning EXIF_YResolution

EXIF Windows XP Tags

EXIF_XPAuthor EXIF_XPComment

EXIF_XPKeywords EXIF_XPRating

EXIF_XPSubject EXIF_XPTitle

EXIF GPS Tags

EXIF_GPSVersionID EXIF_GPSLatitudeRef

EXIF_GPSLatitudeDegrees EXIF_GPSLatitudeMinutes

EXIF_GPSLatitudeSeconds EXIF_GPSLongitudeRef

EXIF_GPSLongitudeDegrees EXIF_GPSLongitudeMinutes

EXIF_GPSLongitudeSeconds EXIF_GPSAltitudeRef

EXIF_GPSAltitude EXIF_GPSTimeStampHour

EXIF_GPSTimeStampMinute EXIF_GPSTimeStampSecond

EXIF_GPSSatellites EXIF_GPSStatus

EXIF_GPSMeasureMode EXIF_GPSDOP

EXIF_GPSSpeedRef EXIF_GPSSpeed

EXIF_GPSTrackRef EXIF_GPSTrack

EXIF GPS Tags

EXIF_GPSImgDirectionRef
EXIF_GPSImgDirection
EXIF_GPSMapDatum
EXIF_GPSDestLatitudeRef
EXIF_GPSDestLatitudeDegrees
EXIF_GPSDestLatitudeMinutes
EXIF_GPSDestLatitudeSeconds
EXIF_GPSDestLongitudeRef
EXIF_GPSDestLongitudeDegrees
EXIF_GPSDestLongitudeMinute
EXIF_GPSDestLongitudeSeconds
EXIF_GPSDestBearingRef
EXIF_GPSDestBearing
EXIF_GPSDestDistanceRef
EXIF_GPSDestDistance
EXIF_GPSDateStamp
Exif_DateToDateTime
EXIF_YCbCrPositioningToString
EXIF_OrientationToString

EXIF Helper Functions

EXIF_ExposureProgramToStr
EXIF_LightSourceToStr
EXIF_FlashToStr
EXIF_MeteringModeToString
EXIF_SensingMethodToString
EXIF_ColorSpaceToString
EXIF_FocalPlaneResolutionUnitToStrin
EXIF_ResolutionUnitToString

PostScript

PS_Compression
PS_PaperHeight
PS_PaperWidth
PS_Title

Unit ImageENIO
TIOParamVals

Adobe PDF

PDF_Author	PDF_Compression	PDF_Creator
PDF_Keywords	PDF_PaperHeight	PDF_PaperWidth
PDF_Producer	PDF_Subject	PDF_Title

RAW (Camera)

RAW_AutoAdjustColors	RAW_BlueScale	RAW_Bright
RAW_Camera	RAW_ExtraParams	RAW_FourColorRGB
RAW_Gamma	RAW_GetExifThumbnail	RAW_HalfSize
RAW_QuickInterpolate	RAW_RedScale	RAW_UseAutoWB
RAW_UseCameraWB		

MediaFile (AVI, MPEG, WMV..)

MEDIAFILE_FrameCount	MEDIAFILE_FrameDelayTime

RealRAW (not camera RAW)

BMPRAW_DataFormat

BMPRAW_ChannelOrder

BMPRAW_HeaderSize

BMPRAW_Planes

BMPRAW_RowAlign

Adobe XMP info

XMP_Info

Unit ImageENIO

TIOParamVals

Methods and Properties

Storage handling

DECLARATION

```
procedure Assign(Source: TIOParamsVals);
```

DESCRIPTION

Assign copies all image format parameters from Source.

Example

```
ImageEnView1.IO.Params.Assign(
ImageEnView2.IO.Params );
```

DECLARATION

```
constructor Create(IEIO: TImageEnIO);
```

DESCRIPTION

A TIOParamsVals is automatically created from TImageEnIO component.

IOParamsVals

DECLARATION

```
procedure LoadFromFile(const FileName:
WideString);
```

DESCRIPTION

Loads the file formats parameters.

Example

```
// Saves default parameters
ImageEnView1.IO.Params.TIFF_Compression:=ioTIFF_LZW;
ImageEnView1.IO.Params.SaveToFile('default');
...
// Load default parameters
ImageEnView1.IO.Params.LoadFromFile('default');
```

Unit ImageENIO
TIOParamVals

DECLARATION

```
procedure LoadFromStream(Stream: TStream);
```

DESCRIPTION

Loads the file formats parameters.

DECLARATION

```
procedure ResetEXIF;
```

DESCRIPTION

This method resets all EXIF fields. This is useful to remove EXIF info from your files. However if you want remove all metadata information from the jpegs we suggest to call ResetInfo instead of ResetEXIF.

DECLARATION

```
procedure ResetInfo;
```

DESCRIPTION

This method resets IPTC information, imaging annotations, jpeg markers, EXIF, GIF comments, PNG comments, TIFF textual tags, TGA comments, PXM comments and any other loaded textual information. It also removes the input ICC profile.

Example

```
//Save jpeg without EXIF or other metadata
//call Params.ResetInfo
ImageEnView1.IO.LoadFromFile('input.jpg');
ImageEnView1.IO.Params.ResetInfo;
ImageEnView1.IO.SaveToFile('output.jpg');
```

DECLARATION

```
procedure SaveToFile(const FileName: WideString);
```

DESCRIPTION

SaveToFile saves the file format parameters.

Example

```
// Saves default parameters
ImageEnView1.IO.Params.TIFF_Compression:=ioTIFF_LZW;
ImageEnView1.IO.Params.SaveToFile('default');
...
// Load default parameters
ImageEnView1.IO.Params.LoadFromFile('default');
```

DECLARATION

```
procedure SaveToStream(Stream: TStream);
```

DESCRIPTION

SaveToStream saves the file format parameters.

DECLARATION

```
procedure SetDefaultParams;
```

DESCRIPTION

Sets default parameters (startup parameters).

DECLARATION

```
procedure UpdateEXIFThumbnail;
```

DESCRIPTION

UpdateEXIFThumbnail updates EXIF_Bitmap property with the content of current image. You should call this method just before save a jpeg to make thumbnail consistent with saved image.

Example

```
ImageEnView1.IO.LoadFromFile('input.jpg');
ImageEnView1.Proc.Negative;
ImageEnView1.IO.Params.UpdateEXIFThumbnail;
ImageEnView1.IO.SaveToFile('output.jpg');
```

Common

DECLARATION

```
property BitsPerSample: integer;
```

DESCRIPTION

BitsPerSample depth is the bits, for each sample.
Allowed values:

Value	Description
1	1 bit black & white
2..7	color mapped bitmap (from 4 to 128 colors)
8	color mapped (256 colors), 8 bit gray scale, 24 bit RGB, 48 bit CMYK, 24 bit CIELab
16	16 bit gray scale or 48 bit RGB
32	32 bit floating point

See also

SamplesPerPixel.

Example

```
// saves 256 colormapped bitmap
ImageEnView1.IO.Params.BitsPerSample:=8;
ImageEnView1.IO.Params.SamplesPerPixel:=1;
ImageEnView1.IO.Params.SaveToFile('alfa.bmp');
```

IOParamsVals

DECLARATION

```
property ColorMapCount: integer;
```

DESCRIPTION

ColorMapCount is the number of entries in the ColorMap array.

Read-only

DECLARATION

```
property ColorMap: PRGBROW;
```

DESCRIPTION

ColorMap gets the colormap of the last loaded image.

Read-only

Example

```
// shows the palette of the file "myfile.gif"
var
fPalDial: TImageEnPaletteDialog;
begin
  ImageEnView1.IO.LoadFromFile('myfile.gif');
  fPalDial := TImageEnPaletteDialog.Create(self);
fPalDial.SetPalette(ImageEnView1.IO.Params.ColorM
ap^,
    ImageEnView1.IO.Params.ColorMapCount);
  fPalDial.Execute;
  fPalDial.free;
end;
```

DECLARATION

```
property DefaultICCProfile: TIEICC;
```

DESCRIPTION

DefaultICCProfile contains a default ICC profile associated with the loaded image. A default ICC profile is used only when the loaded image does not have its own ICC profile.

Example

```
ImageEnView1.IO.Params.DefaultICCProfile.LoadFrom
File('c:\Windows\System32\spool\drivers\color\Ado
beRGB1998.icc');
ImageEnView1.IO.LoadFromFile('input.tif');
```

Unit ImageENIO
TIOParamVals

DECLARATION

```
property Dpi: integer;
```

DESCRIPTION

Dpi specifies the resolution (dots per inch) of the acquired image. If horizontal and vertical resolutions aren't equal, Dpi is the horizontal resolution (the same as DpiX).

See also

DpiX

DpiY

DECLARATION

```
property DpiX: integer;
```

DESCRIPTION

DPIX specifies the DpiX (horizontal dots per inch) of the image.

See also

Dpi

DpiY

DECLARATION

```
property DpiY: integer;
```

DESCRIPTION

DPIY specifies the DpiY (vertical dots per inch) of the image.

See also

Dpi
DpiX

DECLARATION

```
property EnableAdjustOrientation: Boolean;
```

DESCRIPTION

If you set this property to True before load a file which contains EXIF information, the image will be automatically orientated.
This property also sets JPEG_EnableAdjustOrientation and TIFF_EnableAdjustOrientation.

Example

```
ImageEnView1.IO.Params.EnableAdjustOrientation:=true;
ImageEnView1.IO.LoadFromFile('input.jpg');
```

IOParamsVals

Unit ImageENIO

TIOParamVals

DECLARATION

```
property FileName: WideString;
```

DESCRIPTION

Filename of the last loaded or saved image.

Read-only

DECLARATION

```
property FileTypeStr: string;
```

DESCRIPTION

FileTypeStr returns a textual description of the file type contained in the FileType property and other file format specific properties. For example, if FileType is ioTIFF and TIFF_Compression is ioTIFF_G4FAX, FileTypeStr returns the string: 'TIFF Bitmap (TIF) Group 4 Fax'.

Read only

Example

```
if OpenImageEnDialog1.Execute then
begin
ImageEnView1.IO.LoadFromFile(OpenImageEnDialog1.FileName);
ShowMessage( ImageEnView1.IO.Params.FileTypeStr );
end;
```

IOParamsVals

Unit ImageENIO
TIOParamVals

DECLARATION

```
property FileType: TIOFileType;
```

DESCRIPTION

FileType is the last loaded/saved file type. For example, after a call to LoadFromFileGIF, this property will have the ioGIF value.

Example

```
if OpenImageEnDialog1.Execute then begin
 ImageEnView1.IO.LoadFromFile(OpenImageEnDialog1.FileName);
  case ImageEnView1.IO.Params.FileType of
    ioTIFF: ShowMessage('You have loaded a TIFF file');
    ioGIF: ShowMessage('You have loaded a GIF file');
  ...
  end; // case
end;
```

IOParamsVals

Unit ImageENIO
TIOParamVals

DECLARATION

```
property GetThumbnail: Boolean;
```

DESCRIPTION

You have to set this property just before load an image which could contain a thumbnail (like EXIF thumbnail). The thumbnail will be loaded if you set GetThumbnail to true. If no thumbnail is present the full image will be loaded.

Example

```
ImageEnView1.IO.Params.GetThumbnail := true;
ImageEnView1.IO.LoadFromFile('input.jpg');
```

DECLARATION

```
property Height: integer;
```

DESCRIPTION

Height is image height in pixels.

Unit ImageENIO
TIOParamVals

DECLARATION

```
property ImageCount: integer
```

DESCRIPTION

ImageCount returns the number of page/images/frames founds inside the last loaded file. This property is valid for all file formats.

Example

```
ImageEnView1.IO.LoadFromFile('input.tif');
Pages:=ImageEnView1.IO.Params.ImageCount;
```

IOParamsVals

Unit ImageENIO
TIOParamVals

DECLARATION

```
property ImageIndex: integer;
```

DESCRIPTION

ImageIndex specifies the index of next page to load. You have to specify this property just before methods like LoadFromFile. This property is valid for all file formats (TIFF, GIF, DCX, etc..) and sets also other properties like TIFF_ImageIndex, GIF_ImageIndex, etc.

Example

```
ImageEnView1.IO.Params.ImageIndex := 1;
// need second page
ImageEnView1.IO.LoadFromFile('input.tif');
```

IOParamsVals

Unit ImageENIO
TIOParamVals

DECLARATION

```
property ImagingAnnot: TIEImagingAnnot;
```

DESCRIPTION

ImagingAnnot contains the (Wang) imaging annotations loaded (or to save) from a TIFF. Using TIEImagingAnnot object you can create new objects, copy to a TImageEnVect (as vectorial objects), Copy from a TImageEnVect (from vectorial objects) or just draw on the bitmap.

Example

```
// load an image and all annotations from
// input.tif. This allow to modify the
// annotations:
ImageEnVect1.IO.LoadFromFile('input.tif');
ImageEnVect1.IO.Params.ImagingAnnot.CopyToTImageE
nVect( ImageEnVect1 );

// load an image and all annotations from
// input.tif, but just draw on the image (for
// display):
ImageEnVect1.IO.LoadFromFile('input.tif');
ImageEnVect1.IO.Params.ImagingAnnot.DrawToBitmap(
ImageEnVect1.IEBitmap, true );
ImageEnVect1.Update;
```

IOParamsVals

Unit ImageENIO
TIOParamVals

DECLARATION

```
property InputICCProfile: TIEICC;
```

DESCRIPTION

InputICCProfile contains the ICC profile associated with the loaded image. If iegEnableCMS is true and ImageEn has a CMS, the InputICCProfile and OutputICCProfile are applied when you load the image.

Unit ImageENIO
TIOParamVals

DECLARATION

```
property IsResource: Boolean;
```

DESCRIPTION

Some formats (BMP, ICO, CUR), if stored as resources (in EXE, DLL, etc...), do not contain some headers. If you set IsResource=true, these headers are not expected allowing to load correctly the image.

Look at inputoutput/resourceLoader example.

Example

```
// loads resource 143, in "Bitmap"s, from
// "explorer.exe" (should be a little Windows
// logo)
var
re: TIEResourceExtractor;
buffer: pointer;
bufferLen: integer;
begin
  re :=
TIEResourceExtractor.Create('explorer.exe');
try
  buffer := re.GetBuffer('Bitmap',
    'INTRESOURCE:143', bufferLen);
  ImageEnView1.IO.Params.IsResource := true;
  ImageEnView1.IO.LoadFromBuffer(buffer,
  bufferLen, ioBMP);
finally
  re.Free;
end;
end;
```

DECLARATION

```
property OutputICCProfile: TIEICC;
```

DESCRIPTION

OutputICCProfile contains the ICC profile associated with the current display system. The default profile is sRGB. If iegEnableCMS is true and ImageEn has a CMS, the InputICCProfile and OutputICCProfile are applied when you load the image.

DECLARATION

```
property OriginalHeight: integer;
```

DESCRIPTION

OriginalHeight specifies the original height when an image is scaled during loading.

See also
OriginalWidth.

DECLARATION

```
property OriginalWidth: integer;
```

DESCRIPTION

OriginalWidth specifies the original width when an image is scaled during loading.

See also
OriginalHeight.

DECLARATION

```
property SamplesPerPixel: integer;
```

DESCRIPTION

SamplesPerPixel samples (channels) for each pixel. Allowed values:

Parameter	Description
1	Single channel, colormapped or gray scale image
3	Three channels, RGB or CIELab
4	Four channels, CMYK or BGRA

See also
BitsPerSample.

Example

```
// saves 256 colormapped bitmap
ImageEnView1.IO.Params.BitsPerSample := 8;
ImageEnView1.IO.Params.SamplesPerPixel := 1;
ImageEnView1.IO.Params.SaveToFile('alfa.bmp');
```

DECLARATION

```
property Width: integer;
```

DESCRIPTION

Width is the width of the image in pixels.

BMP

DECLARATION

```
property BMP_Compression: TIOBMPCompression;
```

DESCRIPTION

BMP_Compression specifies the compression method.

Only 16 or 256 color bitmap can be saved with RLE compression.

Example

```
// saves compressed bitmap
ImageEnView1.IO.Params.BMP_Compression :=
ioBMP_RLE;
ImageEnView1.IO.SaveToFile('alfa.bmp');
```

DECLARATION

```
property BMP_HandleTransparency: Boolean;
```

DESCRIPTION

BMP file can have 32 bits per pixel. This property controls how interpret the extra byte in the 32 bit word. When BMP_HandleTransparency is true the extra byte is interpreted as alpha channel, otherwise it is just discarded (ignored). This means that when a 32-bit bitmap is displayed in ImageEn, the transparent color is not used and the image is displayed without transparency if the BMP_HandleTransparency property is false. If the BMP_HandleTransparency property is true then the image is displayed with transparency.

Example

```
// BMP_HandleTransparency = true then display
// with transparency
ImageEnVect1.IO.Params.BMP_HandleTransparency :=
BMP_HandleTransparency1.Checked;
ImageEnVect1.IO.LoadFromFile( FilePath );
```

DECLARATION

```
property BMP_Version: TIOBMPVersion;
```

DESCRIPTION

BMP_Version specifies the version of the BMP file.

Example

```
// saves a Windows 3.x bitmap
ImageEnView1.IO.Params.BMP_Version := ioBMP_BM3;
ImageEnView1.IO.SaveToFile('alfa.bmp');
```

GIF

DECLARATION

```
property GIF_Action: TIEGIFAction;
```

DESCRIPTION

GIF_Action specifies how to display the frames.

DECLARATION

```
property GIF_Background: TRGB;
```

DESCRIPTION

GIF_Background is the background color of the GIF. This color should exist in the image.

DECLARATION

```
property GIF_Comments: TStringList;
```

DESCRIPTION

GIF_Comments contains the text comments contained in a GIF file.

Example

```
ImageEnIO1.Params.Gif_Comments.Clear;
ImageEnIO1.Params.GIF_Comments.Add('Hello
world!');
ImageEnIO1.SaveToFie('output.gif');
```

DECLARATION

```
property GIF_DelayTime: integer;
```

DESCRIPTION

GIF_DelayTime is the time (in 1/100 s) the frame remains shown.

DECLARATION

```
property GIF_FlagTranspColor: Boolean;
```

DESCRIPTION

If GIF_FlagTranspColor is true, the GIF_TranspColor color is valid (the image has a transparency color). Note: when the image contains an alpha channel (transparency channel) this property is handled automatically.

Example

```
ImageEnView1.IO.Params.GIF_FlagTranspColor:=True;
ImageEnView1.IO.Params.GIF_TranspColor:=CreateRGB
(0,0,0); // black is the transparent color
```

DECLARATION

```
property GIF_ImageCount: integer;
```

DESCRIPTION

GIF_ImageCount specifies the image count of the last GIF loaded.

Read-only

Example

```
// this returns the frame count of 'animated.gif'
// without load the GIF
ImageEnView1.IO.ParamsFromFile('animated.gif');
ImageCount:=ImageEnView1.IO.Params.GIF_ImageCount;
```

DECLARATION

```
property GIF_ImageIndex: integer;
```

DESCRIPTION

GIF_ImageIndex is the index (from 0) of the image frame.

DECLARATION

```
property GIF_Interlaced: Boolean;
```

DESCRIPTION

If GIF_Interlaced is true, the image is interlaced.

DECLARATION

```
property GIF_Ratio: integer;
```

DESCRIPTION

GIF_Ratio is the Aspect ratio. Width/Height = (Ratio+15)/64

DECLARATION

```
property GIF_TranspColor: TRGB;
```

DESCRIPTION

GIF_TranspColor is the transparency color (If GIF_FlagTranspColor is True). This color should exist in the image.

DECLARATION

```
property GIF_Version: AnsiString;
```

DESCRIPTION

GIF_Version can be ‘GIF89a’ or ‘GIF87a’.

DECLARATION

```
property GIF_WinHeight: integer;
```

DESCRIPTION

GIF_WinHeight is the height of the window where GIF is shown.

DECLARATION

```
property GIF_WinWidth: integer;
```

DESCRIPTION

GIF_WinWidth is the width of the window where GIF is shown.

DECLARATION

```
property GIF_XPos: integer;
```

DESCRIPTION

GIF_XPos is the top-left X coordinate where the frame will be shown.

DECLARATION

```
property GIF_YPos: integer;
```

DESCRIPTION

GIF_YPos is the top-left Y coordinate where the frame will be shown.

JPEG

DECLARATION

```
property JPEG_ColorSpace: TIOJPEGColorSpace;
```

DESCRIPTION

JPEG_ColorSpace specifies the saved/loaded color space. Default is ioJPEG_YCbCr.

Unit ImageENIO

TIOParamVals

DECLARATION

type TIOJPEGColorspace=(ioJPEG_RGB, ioJPEG_GRAYLEV, ioJPEG_YCbCr, ioJPEG_CMYK, ioJPEG_YCbCrK);

DESCRIPTION

Value	Description
ioJPEG_RGB	Separate RGB channels.
ioJPEG_GRAYLEV	Unique intensity channel (gray levels).
ioJPEG_YCbCr	three channels (CCIR Recommendation 601-1)
ioJPEG_CMYK	four channels (Cyan Magenta Yellow Black) - linear conversion
ioJPEG_YCbCrK	four channels (YCbCr and Black)

DECLARATION

```
property JPEG_DCTMethod: TIOJPEGDCTMethod;
```

DESCRIPTION

JPEG_DCTMethod specifies the DCT method.

Unit ImageENIO
TIOParamVals

DECLARATION

type TIOJPEGDctMethod = (ioJPEG_ISLOW, ioJPEG_IFAST, ioJPEG_FLOAT);

DESCRIPTION

Value	Description
ioJPEG_ISLOW	slow but accurate integer algorithm (default).
ioJPEG_IFAST	faster, less accurate integer method.
ioJPEG_FLOAT	floating-point method (machine dependent).

IOParamsVals

DECLARATION

```
property JPEG_CromaSubsampling:
TIOJPEGCromaSubsampling;
```

DESCRIPTION

JPEG_CromaSubsampling specifies the croma subsampling, influencing the jpeg output quality. This property is valid only when JPEG_ColorSpace=ioJPEG_YCbCr.

DECLARATION

```
type TIOJPEGCromaSubsampling = (ioJPEG_MEDIUM,
ioJPEG_HIGH, ioJPEG_NONE);
```

DESCRIPTION

Value	Description
ioJPEG_MEDIUM	4:2:2 croma subsampling. Medium quality.
ioJPEG_HIGH	4:1:1 croma subsampling. Default quality for jpegs. Lowest quality.
ioJPEG_NONE	4:4:4 croma subsampling. Highest quality.

DECLARATION

```
property JPEG_EnableAdjustOrientation: Boolean;
```

DESCRIPTION

If you set this property to true before load a Jpeg which contains EXIF information, the image will be automatically rotated.

Example

```
ImageEnView1.IO.Params.JPEG_EnableAdjustOrientation
:= True;
ImageEnView1.IO.LoadFromFile('input.jpg');
```

DECLARATION

```
property JPEG_GetExifThumbnail: Boolean;
```

DESCRIPTION

Setting this property to true means that you want a thumbnail instead of the full image. This works only if a thumbnail is present in EXIF fields.

Example

```
// I want only the thumbnail
ImageEnView1.IO.Params.JPEG_GetExifThumbnail :=
True;
ImageEnView1.IO.LoadFromFile('input.jpg');
```

Unit ImageENIO
TIOParamVals

DECLARATION

```
property JPEG_MarkerList: TIEMarkerList;
```

DESCRIPTION

JPEG_MarkerList contains a list of markers contained in a JPEG file. JPEG markers can contain text, objects and images. Applications can read/write raw markers from JPEG_MarkerList using stream or memory buffer. JPEG_MarkerList contains markers from APP0 to APP15 and COM.

Example

```
// read the JPEG_COM marker (idx is integer,
// Comment is string)
ImageEnView1.IO.LoadFromFile('image.jpg');
Idx :=
ImageEnView1.IO.Params.JPEG_MarkerList.IndexOf(JP
EG_COM);
Comment :=
ImageEnView1.IO.Params.JPEG_MarkerList.MarkerData
[idx];
// now write the JPEG_COM marker
Comment := 'This is the new comment';
ImageEnView1.IO.Params.JPEG_MarkerList.SetMarker(
idx,JPEG_COM,PAnsiChar(Comment),length(Comment));
ImageEnView1.IO.SaveToFile('image.jpg');
```

IOParamsVals

Declaration

```
property JPEG_OptimalHuffman: Boolean;
```

Description

If True, specifies the Jpeg compressor to use an optimal Huffman table (more compression). If false, a standard table is used.

Declaration

```
property JPEG_OriginalWidth: integer;
```

Description

JPEG_OriginalWidth specifies the original width when the jpeg image is scaled (using JPEG_Scale). JPEG_OriginalWidth contains the same value of OriginalWidth. See also JPEG_OriginalHeight.

Declaration

```
property JPEG_OriginalHeight: integer;
```

Description

JPEG_OriginalHeight specifies the original height when the Jpeg image is scaled (using JPEG_Scale). JPEG_OriginalHeight contains the same value of OriginalHeight. See also JPEG_OriginalWidth.

Unit ImageENIO
TIOParamVals

DECLARATION

```
property JPEG_Progressive: Boolean;
```

DESCRIPTION

JPEG_Progressive specifies if this is a progressive jpeg.

IOParamsVals

Unit ImageENIO
TIOParamVals

DECLARATION

```
property JPEG_Quality: integer;
```

DESCRIPTION

JPEG_Quality is the quality factor, from 1 to 100. High values produce better image quality but require more disk space.

It is possible to estimate the original jpeg compression using IECalcJpegStreamQuality or IECalcJpegFileQuality function.

Example

Jpeg is a file format with variable compression rate. The property that regulates the compression (and the quality) is JPEG_Quality. So you should set this property before save

```
// compress my JPEGs to make them smaller
ImageEnView1.IO.LoadFromFile('input.jpg');
ImageEnView1.IO.Params.JPEG_Quality:=70;
ImageEnView1.IO.SaveToFile('output.jpg');
```

The default is 80, while other software uses 70. If you want estimate the value used to create your file, execute this:

```
quality := IECalcJpegFileQuality('input.jpg');
```

```
So you could write:
```

```
ImageEnView1.IO.LoadFromFile('input.jpg');
ImageEnView1.IO.Params.JPEG_Quality :=
IECalcJpegFileQuality('input.jpg');
ImageEnView1.IO.SaveToFile('output.jpg');
```

IOParamsVals

If this is not enough, probably the file contains metatags (text info). To remove them execute:

```
 ImageEnView1.IO.Params.ResetInfo;
//just before saving the file
```

DECLARATION

```
property JPEG_Scale_Used: integer;
```

DESCRIPTION

JPEG_Scale_Used contains the denominator of the scale used to load last Jpeg image. It is used only when JPEG_Scale is ioJPEG_AUTOCALC.

Example

```
ImageEnView1.IO.Params.Width:=100;
ImageEnView1.IO.Params.Height:=100;
ImageEnView1.IO.Params.JPEG_Scale:=ioJPEG_AUTOCAL
C;
ImageEnView1.IO.LoadFromFile('my.jpg');
case ImageEnView1.IO.Params.JPEG_Scale_Used of
  2: ShowMessage('Half');
  4: ShowMessage('Quarter');
  8: ShowMessage('HEIGHTH');
end;
```

IOParamsVals

Unit ImageENIO
TIOParamVals

DECLARATION

```
property JPEG_Scale: TIOJPEGScale;
```

DESCRIPTION

JPEG_Scale defines the size of a loaded JPEG image. This property affects the speed of load.

Example

```
// This is the fastest way to load a jpeg
// thumbnail jpeg of about 100x100 pixels
ImageEnView1.IO.Params.Width := 100;
ImageEnView1.IO.Params.Height := 100;
ImageEnView1.IO.Params.JPEG_Scale :=
ioJPEG_AUTOCALC;
ImageEnView1.IO.LoadFromFile('my.jpg');
```

Unit ImageENIO

TIOParamVals

Declaration

```
TIOJPEGScale = (
ioJPEG_AUTOCALC,
ioJPEG_FULLSIZE,
ioJPEG_HALF,
ioJPEG_QUARTER,
ioJPEG_EIGHTH);
```

Description

Value	Description
ioJPEG_AUTOCALC	self calculates JPEG_Scale to match Width and Height properties;
ioJPEG_FULLSIZE	full size image (default). Slow loading.
ioJPEG_HALF	½ of the full size;
ioJPEG_QUARTER	¼ of the full size;
ioJPEG_EIGHTH	1/8 of the full size. Very fast loading.

Unit ImageENIO
TIOParamVals

DECLARATION

```
property JPEG_Smooth: integer;
```

DESCRIPTION

Smoothing factor (0 is none, 100 is max). If JPEG_Smooth is not zero, the jpeg compressor smooth the image before compress it. This improves compression.

Unit ImageENIO
TIOParamVals

DECLARATION

```
property JPEG_WarningCode: integer;
```

DESCRIPTION

JPEG_WarningCode is the last warning code encountered during loading a Jpeg.
0 : ARITH_NOTIMPL ‘There are legal restrictions on arithmetic coding’
1 : BAD_ALIGN_TYPE ‘ALIGN_TYPE is wrong’
2 : BAD_ALLOC_CHUNK “MAX_ALLOC_CHUNK is wrong”
3 : BAD_BUFFER_MODE “Bogus buffer control mode”
4 : BAD_COMPONENT_ID “Invalid component ID in SOS”
5 : BAD_DCT_COEF “DCT coefficient out of range”
6 : BAD_DCTSIZE “IDCT output block size not supported”
7 : BAD_HUFF_TABLE “Bogus Huffman table definition”
8 : BAD_IN_COLORSPACE “Bogus input colorspace”
9 : BAD_J_COLORSPACE “Bogus JPEG colorspace”
10 : BAD_LENGTH “Bogus marker length”
11 : BAD_LIB_VERSION ‘Wrong JPEG library version’
12 : BAD_MCU_SIZE “Sampling factors too large for interleaved scan”
13 : BAD_POOL_ID “Invalid memory pool code”
14 : BAD_PRECISION “Unsupported JPEG data precision”
15 : BAD_PROGRESSION “Invalid progressive parameters”
16 : BAD_PROG_SCRIPT “Invalid progressive parameters”
17 : BAD_SAMPLING “Bogus sampling factors”
18 : BAD_SCAN_SCRIPT “Invalid scan script”
19 : BAD_STATE “Improper call to JPEG library”
20 : BAD_STRUCT_SIZE “JPEG parameter struct mismatch “
21 : BAD_VIRTUAL_ACCESS “Bogus virtual array access”
22 : BUFFER_SIZE “Buffer passed to JPEG library is too small”

IOParamsVals

23 : CANT_SUSPEND “Suspension not allowed here”
24 : CCIR601_NOTIMPL “CCIR601 sampling not implemented yet”
25 : COMPONENT_COUNT “Too many color components “
26 : CONVERSION_NOTIMPL “Unsupported color conversion request”
27 : DAC_INDEX “Bogus DAC “
28 : DAC_VALUE “Bogus DAC “
29 : DHT_INDEX “Bogus DHT “
30 : DQT_INDEX “Bogus DQT “
31 : EMPTY_IMAGE “Empty JPEG image (DNL not supported)”
32 : EMS_READ “Read from EMS failed”
33 : EMS_WRITE “Write to EMS failed”
34 : EOI_EXPECTED “Didn’t expect more than one scan”
35 : FILE_READ “Input file read error”
36 : FILE_WRITE “Output file write error --- out of disk space?”
37 : FRACT_SAMPLE_NOTIMPL “Fractional sampling not implemented yet”
38 : HUFF_CLEN_OVERFLOW “Huffman code size table overflow”
39 : HUFF_MISSING_CODE “Missing Huffman code table entry”
40 : IMAGE_TOO_BIG “Image too big’
41 : INPUT_EMPTY “Empty input file”
42 : INPUT_EOF “Premature end of input file”
43 : MISMATCHED_QUANT_TABLE “Cannot transcode due to multiple use of quantization table”
44 : MISSING_DATA “Scan script does not transmit all data”
45 : MODE_CHANGE “Invalid color quantization mode change”
46 : NOTIMPL “Not implemented yet”
47 : NOT_COMPILED “Requested feature was omitted at compile time”
48 : NO_BACKING_STORE “Backing store not supported”
49 : NO_HUFF_TABLE “Huffman table was not defined”
50 : NO_IMAGE “JPEG datastream contains no image”

IOParamsVals

51 : NO_QUANT_TABLE “Quantization table was not defined”
52 : NO_SOI “Not a Jpeg file”
53 : OUT_OF_MEMORY “Insufficient memory”
54 : QUANT_COMPONENTS “Cannot quantize”
55 : QUANT_FEW_COLORS “Cannot quantize”
56 : QUANT_MANY_COLORS “Cannot quantize”
57 : SOF_DUPLICATE “Invalid Jpeg file structure: two SOF markers”
58 : SOF_NO_SOS “Invalid Jpeg file structure: missing SOS marker”
59 : SOF_UNSUPPORTED “Unsupported JPEG process”
60 : SOI_DUPLICATE “Invalid Jpeg file structure: two SOI markers”
61 : SOS_NO_SOF “Invalid Jpeg file structure: SOS before SOF”
62 : TFILE_CREATE “Failed to create temporary file”
63 : TFILE_READ “Read failed on temporary file”
64 : TFILE_SEEK “Seek failed on temporary file”
65 : TFILE_WRITE “Write failed on temporary file --- out of disk space?”
66 : TOO_LITTLE_DATA “Application transferred too few scanlines”
67 : UNKNOWN_MARKER “Unsupported marker”
68 : VIRTUAL_BUG “Virtual array controller messed up”
69 : WIDTH_OVERFLOW “Image too wide for this implementation”
70 : XMS_READ “Read from XMS failed”
71 : XMS_WRITE “Write to XMS failed”
72 : COPYRIGHT
73 : VERSION
74 : 16BIT_TABLES “Caution: quantization tables are too coarse for baseline JPEG”
75 : ADOBE “Adobe APP14 marker”
76 : APP0 “Unknown APP0 marker”
77 : APP14 “Unknown APP14 marker”
78 : DAC “Define Arithmetic Table”
79 : DHT “Define Huffman Table”

80 : DQT “Define Quantization Table”
81 : DRI “Define Restart Interval”
82 : EMS_CLOSE “Freed EMS handle
83 : EMS_OPEN “Obtained EMS handle”
84 : EOI “End Of Image”
85 : HUFFBITS
86 : JFIF “JFIF APP0 marker “
87 : JFIF_BADTHUMBNAILSIZE “Warning: thumbnail image size does not match data length”
88 : JFIF_EXTENSION “JFIF extension marker”
89 : JFIF_THUMBNAIL
90 : MISC_MARKER “Miscellaneous marker”
91 : PARMLESS_MARKER “Unexpected marker”
92 : QUANTVALS
93 : QUANT_3_NCOLORS
94 : QUANT_NCOLORS
95 : QUANT_SELECTED
96 : RECOVERY_ACTION
97 : RST
98 : SMOOTH_NOTIMPL “Smoothing not supported with nonstandard sampling ratios”
99 : SOF
100 : SOF_COMPONENT
101 : SOI “Start of Image”
102 : SOS “Start Of Scan”
103 : SOS_COMPONENT
104 : SOS_PARAMS
105 : TFILE_CLOSE “Closed temporary file”
106 : TFILE_OPEN “Opened temporary file”
107 : THUMB_JPEG “JFIF extension marker “

IOParamsVals

108 : THUMB_PALETTE “JFIF extension marker “
109 : THUMB_RGB “JFIF extension marker “
110 : UNKNOWN_IDS “Unrecognized component IDs, assuming YCbCr”
111 : XMS_CLOSE “Freed XMS handle”
112 : XMS_OPEN “Obtained XMS handle”
113 : ADOBE_XFORM “Unknown Adobe color transform code”
114 : BOGUS_PROGRESSION “Inconsistent progression sequence “
115 : EXTRANEOUS_DATA “Corrupt JPEG data: extraneous bytes before marker”
116 : HIT_MARKER “Corrupt JPEG data: premature end of data segment”
117 : HUFF_BAD_CODE “Corrupt JPEG data: bad Huffman code”
118 : JFIF_MAJOR “Warning: unknown JFIF revision number”
119 : JPEG_EOF “Premature end of Jpeg file”
120 : MUST_RESYNC “Corrupt JPEG data “
121 : NOT_SEQUENTIAL “Invalid SOS parameters for sequential JPEG”
122 : TOO_MUCH_DATA “Application transferred too many scanlines”

DECLARATION

```
property JPEG_WarningTot: integer;
```

DESCRIPTION

JPEG_WarningTot is the count of all warnings encountered during loading a Jpeg. We suggest use of the Aborting property to see if the image is corrupted.

JPEG 2000

DECLARATION

```
property J2000_ColorSpace: TIOJ2000ColorSpace;
```

DESCRIPTION

J2000_ColorSpace specifies the saved/loaded color space.

DECLARATION

```
property J2000_Rate: double;
```

DESCRIPTION

J2000_Rate specifies the compression rate to apply. Allowed values from 0 to 1, where 1 is no compression (lossless mode).

Example

```
// saves lossless
ImageEnIO1.Params.J2000_Rate := 1;
ImageEnIO1.SaveToFile('output.jp2');

// saves with loss compression
ImageEnView1.IO.Params.J2000_Rate := 0.015;
ImageEnView1.IO.SaveToFile('output.jp2');
```

Unit ImageENIO
TIOParamVals

DECLARATION

```
property J2000_ScalableBy: TIOJ2000ScalableBy;
```

DESCRIPTION

J2000_ScalableBy specifies the progression order.

DECLARATION

```
TIOJ2000ScalableBy = (
ioJ2000_Rate,
ioJ2000_Resolution);
```

DESCRIPTION

Value	Description
ioJ2000_Rate	layer-resolution-component-position (LRCP) progressive (i.e., rate scalable)
ioJ2000_Resolution	resolution-layer-component-position (RLCP) progressive (i.e., resolution scalable)

PCX

DECLARATION

```
property PCX_Compression: TIOPCXCompression;
```

DESCRIPTION

PCX_Compression is the compression type for PCX image format.

DECLARATION

```
type TIOPCXCompression = (ioPCX_UNCOMPRESSED,
ioPCX_RLE);
```

DESCRIPTION

ioPCX_UNCOMPRESSED is uncompressed PCX (incompatible with most PCX readers). ioPCX_RLE is compressed PCX (standard PCX).

DECLARATION

```
property PCX_Version: integer;
```

DESCRIPTION

PCX_Version (5 is the default).

Adobe PSD

Declaration

```
property PSD_LoadLayers: Boolean;
```

Description

If True discards the merged image and load only separated layers. If PSD_LoadLayers is false (default) only the merged image is loaded.

Example

```
// loads a multilayer PSD and allow user to move
and resize layers
ImageEnView1.IO.Params.PSD_LoadLayers := true;
ImageEnView1.IO.LoadFromFile('input.psd');
ImageEnView1.MouseInteract := [ miMoveLayers,
miResizeLayers];
```

Declaration

```
property PSD_ReplaceLayers: Boolean;
```

Description

If true current layers are replaced by the content of PSD file, otherwise the PSD file content is append to existing layers.

DECLARATION

```
property PSD_HasPremultipliedAlpha: Boolean;
```

DESCRIPTION

If PSD_HasPremultipliedAlpha is true the alpha channel is premultiplied.

Read-only.

HDP

DECLARATION

```
property HDP_ImageQuality: double;
```

DESCRIPTION

0.0 specifies the lowest possible fidelity rendition and 1.0 specifies the highest fidelity, which for Microsoft HD Photo results in mathematically lossless compression. The default value is 0.9.

DECLARATION

```
property HDP_Lossless: Boolean;
```

DESCRIPTION

Setting this parameter to true enables mathematically lossless compression mode and overrides the HDP_ImageQuality parameter setting. The default value is false.

TIFF

DECLARATION

```
property TIFF_ByteOrder: TIOByteOrder;
```

DESCRIPTION

TIFF_ByteOrder specifies the byte order of the last loaded image. This property is readonly, because ImageEn uses always ioLittleEndian when saves TIFF.

ioBigEndian byte order is used by Motorola and PowerPC (non- Intel Macintosh), while ioLittleEndian is used by Intel processors.

This function is useful, for example, in InsertTIFFImageFile and InsertTIFFImageStream because this function just merges the two tiff images, without decoding them. If the images have different byte order the resulting TIFF is un-readable.

Unit ImageENIO

TIOParamVals

Declaration

```
TIOByteOrder = (ioBigEndian, ioLittleEndian); //
ioBigEndian=Motorola, ioLittleEndian=Intel
```

Declaration

```
property TIFF_Compression: TIOTIFFCompression;
```

Description

TIFF_Compression specifies the compression type for TIFF image format.

Declaration

```
type TIOTIFFCompression = (ioTIFF_UNCOMPRESSED,
ioTIFF_CCITT1D, ioTIFF_G3FAX1D, ioTIFF_G3FAX2D,
ioTIFF_G4FAX, ioTIFF_LZW, ioTIFF_OLDJPEG,
ioTIFF_JPEG, ioTIFF_PACKBITS, ioTIFF_ZIP,
ioTIFF_DEFLATE, ioTIFF_UNKNOWN);
```

IOParamsVals

Unit ImageENIO
TIOParamVals

DESCRIPTION

Value	Description
ioTIFF_UNCOMPRESSED	Uncompressed TIFF. Supports all pixel formats and alpha channel.
ioTIFF_CCITT1D	Bilevel Huffman compression. Only black/white images, no alpha channel.
ioTIFF_G3FAX1D	Bilevel Group 3 CCITT compression, monodimensional. Only black/white images, no alpha channel.
ioTIFF_G3FAX2D	Bilevel Group 3 CCITT compression, bidimensional. Only black/white images, no alpha channel.
ioTIFF_G4FAX	Bilevel Group 4 CCITT compression, bidimensional. Only black/white images, no alpha channel.
IoTIFF_LZW	LZW compression. Supports alpha channel.
ioTIFF_OLDJPEG	Ver.6.0 jpeg compression (unsupported). Only true color images, no alpha channel.
ioTIFF_JPEG	Jpeg compression. Only true color images, no alpha channel.
ioTIFF_PACKBITS	RLE compression. Supports alpha channel.
ioTIFF_ZIP	ZIP compression (non-TIFF standard). No alpha channel.

Value	Description
ioTIFF_DEFLATE	Adobe ZIP compression (non-TIFF standard). No alpha channel.
ioTIFF_UNKNOWN	Unknown compression

DECLARATION

```
property TIFF_DocumentName: AnsiString;
```

DESCRIPTION

TIFF_DocumentName specifies the document name field of a TIFF image format.

Example

```
ImageEnView1.IO.Params.TIFF_DocumentName :=
'Sanremo';
ImageEnView1.IO.Params.TIFF_ImageDescription :=
'The city of the flowers';
ImageEnView1.IO.SaveToFile('Italy.tif');
```

IOParamsVals

DECLARATION

```
property TIFF_EnableAdjustOrientation: Boolean;
```

DESCRIPTION

When TIFF_EnableAdjustOrientation is True, ImageEn rotates loaded images as specified in the TIFF file. (The default value is false).

DECLARATION

```
property TIFF_GetTile: integer;
```

DESCRIPTION

TIFF_GetTile specifies the tile number to load when loading a tiled TIFF file. -1 (default) means “load all tiles”.

DECLARATION

```
property TIFF_FillOrder: integer;
```

DESCRIPTION

TIFF_FillOrder specifies the logical order of bits within a byte.
1 = pixels are arranged within a byte such that pixels with lower column values are stored in the higher-order bits of the byte.
2 = pixels are arranged within a byte such that pixels with lower column values are stored in the lower-order bits of the byte.

DECLARATION

```
property TIFF_ImageCount: integer;
```

DESCRIPTION

TIFF_ImageCount specifies the image count of the last TIFF loaded.

Read-only

Example

```
// this return the frame count of 'pages.tiff'
// without load the TIFF
ImageEnView1.IO.ParamsFromFile('pages.tiff');
ImageCount:=ImageEnView1.IO.Params.TIFF_ImageCount;
```

IOParamsVals

DECLARATION

```
property TIFF_ImageDescription: AnsiString;
```

DESCRIPTION

TIFF_ImageDescription specify the image description field of a TIFF image format.

Example

```
ImageEnView1.IO.Params.TIFF_DocumentName :=
'Sanremo';
ImageEnView1.IO.Params.TIFF_ImageDescription :=
'The city of the flowers';
ImageEnView1.IO.SaveToFile('Italy.tif');
```

DECLARATION

```
property TIFF_ImageIndex: integer;
```

DESCRIPTION

TIFF_ImageIndex is the index (from 0) of the image.

DECLARATION

```
property TIFF_JPEGColorSpace: TIOJPEGColorSpace;
```

DESCRIPTION

TIFF_JPEGColorSpace specifies the desired color space for the Jpeg included in a TIFF file.

Example

```
ImageEnView1.IO.Params.TIFF_JPEGColorSpace :=
ioJPEG_RGB;
ImageEnView1.IO.Params.TIFF_Compression :=
ioTIFF_JPEG;
ImageEnView.IO.SaveToFile('output.tif');
```

IOParamsVals

DECLARATION

```
type TIOJPEGColorspace=(ioJPEG_RGB,
ioJPEG_GRAYLEV, ioJPEG_YCbCr, ioJPEG_CMYK,
ioJPEG_YCbCrK);
```

DESCRIPTION

Value	Description
ioJPEG_RGB	separate RGB channels.
ioJPEG_GRAYLEV	unique intensity channel (gray levels).
ioJPEG_YCbCr	three channels (CCIR Recommendation 601-1)
ioJPEG_CMYK	four channels (Cyan Magenta Yellow Black) - linear conversion
ioJPEG_YCbCrK	four channels (YCbCr and Black)

DECLARATION

```
property TIFF_JPEGQuality: integer;
```

DESCRIPTION

TIFF_JPEGQuality is a quality factor, from 1 to 100 for TIFF compressed as JPEG. High values produce better image quality but require more disk space.

IOParamsVals

Unit ImageENIO
TIOParamVals

DECLARATION

```
property TIFF_Orientation: integer;
```

DESCRIPTION

The TIFF_Orientation is the orientation of the original image. ImageEn adjusts automatically the image if TIFF_EnableAdjustOrientation is true. Allowed values are:

Value	Description
1	The 0th row represents the visual top of the image, and the 0th column represents the visual left-hand side.
2	The 0th row represents the visual top of the image, and the 0th column represents the visual right-hand side.
3	The 0th row represents the visual bottom of the image, and the 0th column represents the visual right-hand side.
4	The 0th row represents the visual bottom of the image, and the 0th column represents the visual left-hand side.
5	The 0th row represents the visual left-hand side of the image, and the 0th column represents the visual top.
6	The 0th row represents the visual right-hand side of the image, and the 0th column represents the visual top.
7	The 0th row represents the visual right-hand side of the image, and the 0th column represents the visual bottom.
8	The 0th row represents the visual left-hand side of the image, and the 0th column represents the visual bottom.

Support for orientations other than 1 is not a baseline TIFF requirement.

Read-only

DECLARATION

```
property TIFF_PageCount: integer;
```

DESCRIPTION

TIFF_PageCount is the number of pages in a TIFF file.

DECLARATION

```
property TIFF_PageName: AnsiString;
```

DESCRIPTION

TIFF_PageName is the page name field in a TIFF image format.

DECLARATION

```
property TIFF_PageNumber: integer;
```

DESCRIPTION

TIFF_PageNumber is the page number.

Unit ImageENIO

TIOParamVals

Declaration

```
property TIFF_PhotometInterpret:
TIOTIFFPhotometInterpret;
```

Description

Photometric interpretation.

Declaration

```
type TIOTIFFPhotometInterpret =
(ioTIFF_WHITEISZERO, ioTIFF_BLACKISZERO,
ioTIFF_RGB, ioTIFF_RGBPALETTE, ioTIFF_TRANSPMASK,
ioTIFF_CMYK, ioTIFF_YCBCR, ioTIFF_CIELAB);
```

DECLARATION

```
property TIFF_PlanarConf: integer;
```

DESCRIPTION

TIFF_PlanarConf is the planar configuration.

DECLARATION

```
property TIFF_SubIndex: integer;
```

DESCRIPTION

A TIFF can contain several pages. You can load image or parameters from a page using TIFF_ImageIndex (or just ImageIndex). Each page can contain sub images, like a tree, so you can load a specific subpage using TIFF_SubIndex. This is useful for example to load an embedded Jpeg from a DNG (camera raw format), that is located at first page and second sub index.

Example

```
// loads the embeded jpeg from a DNG
ImageEnView1.IO.Params.TIFF_ImageIndex := 0;
ImageEnView1.IO.Params.TIFF_SubIndex := 1;
ImageEnView1.IO.LoadFromFileTIFF('DCSXXX.DNG');
```

DECLARATION

```
property TIFF_XPos: integer;
```

DESCRIPTION

X top-left position of the original scanned image.

DECLARATION

```
property TIFF_YPos: integer;
```

DESCRIPTION

Y top-left position of the original scanned image.

DECLARATION

```
property TIFF_ZIPCompression: integer;
```

DESCRIPTION

Specifies the compression level to be used when compression is ioTIFF_ZIP.
0 = fastest
1 = normal (default)
2 = max

CUR

DECLARATION

```
property CUR_Background: TRGB;
```

DESCRIPTION

CUR_Background is the background color where to merge loaded image. Applications should set this property before loading images.

Example

```
ImageEnView1.IO.Params.CUR_Background :=
CreateRGB(255,255,255);
ImageEnView1.IO.LoadFromFileCUR('myicon.cur');
```

DECLARATION

```
property CUR_ImageIndex: integer;
```

DESCRIPTION

The CUR_ImageIndex is the index (from 0) of the image.
A CUR file can contain multiple images inside.

Example

```
// Load the second cursor inside 'alfa.cur'
ImageEnView1.IO.Params.CUR_ImageIndex := 1;
ImageEnView1.IO.LoadFromFile('alfa.cur');
```

IOParamsVals

Unit ImageENIO
TIOParamVals

DECLARATION

```
property CUR_XHotSpot: integer;
```

DESCRIPTION

CUR_XHotSpot is the horizontal position of hot spot point.
See also CUR_XHotSpot.

DECLARATION

```
property CUR_YHotSpot: integer;
```

DESCRIPTION

CUR_YHotSpot is the vertical position of hot spot point.
See also CUR_XHotSpot.

IOParamsVals

DCX

DECLARATION

```
property DCX_ImageIndex: integer;
```

DESCRIPTION

Specifies the image index (0=first page) to load for a DCX file.

Example

```
// I want the second page of 'input.dcx'
ImageEnView1.IO.Params.DCX_ImageIndex := 1;
ImageEnView1.IO.LoadFromFile('input.dcx');
```

ICO

DECLARATION

```
property ICO_background: TRGB;
```

DESCRIPTION

ICO_background is the background color where to merge loaded image. Applications should set this property before loading images. ICO_Background is used only when you load icon files.
This is needed to set a specific color as transparent (the image background) for viewing.

When you save an image as an icon, only the image alpha channel is used.

If you want to make a color transparent you should use SetTransparentColors before saving the file.

Reading example

```
ImageEnView1.IO.Params.ICO_Background :=
CreateRGB(255,255,255);
ImageEnView1.IO.LoadFromFileICO('myicon.ico');
```

Writing example

```
ImageEnView1.Proc.SetTransparentColors(
  transparent_color, transparent_color, 0 );
ImageEnView1.IO.SaveToFileICO('myicon.ico');
```

IOParamsVals

DECLARATION

```
property ICO_BitCount: TIOICOBitCount;
```

DESCRIPTION

ICO_BitCount is an array of integers. It contains the input/output bitcount of the images contained in an ICO file.
The last bitcount must contain 0.

Example

```
// save the current image in 'output.ico', It
will contain three images with 64x64 32bit (24bit
+alphachannel), 32x32 256 colors and 32x32 16
colors
// 64x64 x32bit
ImageEnView.IO.Params.ICO.BitCount[0]:=32;
ImageEnView.IO.Params.ICO.Sizes[0].cx:=64;
ImageEnView.IO.Params.ICO.Sizes[0].cy:=64;
// 32x32 x8bit
ImageEnView.IO.Params.ICO.BitCount[1]:=8;
ImageEnView.IO.Params.ICO.Sizes[1].cx:=32;
ImageEnView.IO.Params.ICO.Sizes[1].cy:=32;
// 32x32 x4bit
ImageEnView.IO.Params.ICO.BitCount[2]:=4;
ImageEnView.IO.Params.ICO.Sizes[2].cx:=32;
ImageEnView.IO.Params.ICO.Sizes[2].cy:=32;
// I don't want other images
ImageEnView.IO.Params.ICO.BitCount[3]:=0;
ImageEnView.IO.Params.ICO.Sizes[3].cx:=0;
ImageEnView.IO.Params.ICO.Sizes[3].cy:=0;
// save
ImageEnView.IO.SaveToFile('output.ico');
```

IOParamsVals

Unit ImageENIO

TIOParamVals

DECLARATION

```
TIOICOBitCount = array[0..IEMAXICOIMAGES - 1] of
integer;
```

DECLARATION

```
property ICO_ImageIndex: integer;
```

Example

```
// Load the second image inside 'alfa.ico'
ImageEnView1.IO.Params.ICO_ImageIndex := 1;
ImageEnView1.IO.LoadFromFile('alfa.ico');
```

IOParamsVals

DESCRIPTION

The ICO_ImageIndex is the index (from 0) of the image. An ICO file can contain multiple images inside.

DECLARATION

```
property ICO_Sizes: TIOICOSizes;
```

Example

```
// save the current image in 'output.ico', It
will contain three images with 64x64 32bit (24bit
+alphachannel), 32x32 256 colors and 32x32 16
colors
// 64x64 x32bit
ImageEnView.IO.Params.ICO.BitCount[0]:=32;
ImageEnView.IO.Params.ICO.Sizes[0].cx:=64;
ImageEnView.IO.Params.ICO.Sizes[0].cy:=64;
// 32x32 x8bit
ImageEnView.IO.Params.ICO.BitCount[1]:=8;
ImageEnView.IO.Params.ICO.Sizes[1].cx:=32;
ImageEnView.IO.Params.ICO.Sizes[1].cy:=32;
// 32x32 x4bit
ImageEnView.IO.Params.ICO.BitCount[2]:=4;
ImageEnView.IO.Params.ICO.Sizes[2].cx:=32;
ImageEnView.IO.Params.ICO.Sizes[2].cy:=32;
// I don't want other images
ImageEnView.IO.Params.ICO.BitCount[3]:=0;
ImageEnView.IO.Params.ICO.Sizes[3].cx:=0;
ImageEnView.IO.Params.ICO.Sizes[3].cy:=0;
// save
ImageEnView.IO.SaveToFile('output.ico');
```

IOParamsVals

DESCRIPTION

ICO_Sizes is an array of TSize structures. It contains the input/output sizes of the images contained in an ICO file.
The last size must contain size 0.

DECLARATION

```
TIOICOSizes = array[0..IEMAXICOIMAGES - 1] of TSize;
```

PNG

DECLARATION

```
property PNG_Background: TRGB;
```

DESCRIPTION

PNG_Background specifies the background color of the image.

DECLARATION

```
property PNG_Compression: integer;
```

DESCRIPTION

PNG_Compression changes how much time PNG-compressor spends on trying to compress the image data. This property accepts values from 0 (no compression) to 9 (best compression).

Example

```
// Set best compression
ImageEnView1.IO.Params.PNG_Filter:=ioPNG_FILTER_P
AETH;
ImageEnView1.IO.Params.PNG_Compression := 9;
// Save PNG
ImageEnView1.IO.SaveToFilePNG('max.png');
```

DECLARATION

```
property PNG_Filter: TIOPNGFilter;
```

DESCRIPTION

Set the PNG_Filter filter to apply. This can have a significant impact on the size and encoding speed and a somewhat lesser impact on the decoding speed of an image.

Example

```
// Set best compression
ImageEnView1.IO.Params.PNG_Filter:=ioPNG_FILTER_P
AETH;
ImageEnView1.IO.Params.PNG_Compression := 9;
// Save PNG
ImageEnView1.IO.SaveToFilePNG('max.png');
```

IOParamsVals

TIOParamVals

DECLARATION

```
TIOPNGFilter = (
ioPNG_FILTER_NONE,
ioPNG_FILTER_SUB,
ioPNG_FILTER_PAETH,
ioPNG_FILTER_UP,
ioPNG_FILTER_AVG,
ioPNG_FILTER_ALL);
```

DECLARATION

```
property PNG_Interlaced: Boolean;
```

DESCRIPTION

The image is interlaced if PNG_Interlaced is True.

Unit ImageENIO
TIOParamVals

DECLARATION

```
property PNG_TextKeys: TStringList;
```

DESCRIPTION

Contains a list of keys associated to a list of values (PNG_TextValues property). PNG_TextKeys and PNG_TextValues must contain the same number of values. Only uncompressed text is supported.

Example

```
// add author and other text info in a PNG file
ImageEnView1.IO.Params.PNG_TextKeys.Add('Author');
ImageEnView1.IO.Params.PNG_TextValues.Add('Letizia');
ImageEnView1.IO.Params.PNG_TextKeys.Add('Subject');
ImageEnView1.IO.Params.PNG_TextValues.Add('Colosseo')
;
ImageEnView1.IO.SaveToFile('output.png');

// read all text info from a PNG
for i:=0 to
ImageEnView1.IO.Params.PNG_TextKeys.Count-1 do
begin
  key := ImageEnView1.IO.Params.PNG_TextKeys[i];
  value := ImageEnView1.IO.Params.PNG_TextValues[I];
end;
```

IOParamsVals

DECLARATION

```
property PNG_TextValues: TStringList;
```

DESCRIPTION

Contains a list of values associated to a list of keys (PNG_TextKeys property). PNG_TextKeys and PNG_TextValues must contain the same number of values. Only uncompressed text is supported.

TGA

DECLARATION

```
property TGA_AspectRatio: double;
```

DESCRIPTION

TGA_AspectRatio contains the pixel aspect ratio (pixel width/height) of the TGA image.

DECLARATION

```
property TGA_Author: AnsiString;
```

DESCRIPTION

TGA_Author contains the author name of the TGA file.

DECLARATION

```
property TGA_Background: TRGB;
```

DESCRIPTION

TGA_Background contains the image background (or transparency) color.

DECLARATION

```
property TGA_Compressed: Boolean;
```

DESCRIPTION

Set the TGA_Compressed property to True to compress a TGA image (RLE compression).

DECLARATION

```
property TGA_Date: TDateTime;
```

DESCRIPTION

TGA_Date contains the date/time creation of the TGA file.

DECLARATION

```
property TGA_Descriptor: AnsiString;
```

DESCRIPTION

The TGA_Descriptor contains the descriptor of the TGA file.

DECLARATION

```
property TGA_Gamma: double;
```

DESCRIPTION

TGA_Gamma contains the gamma value of the TGA image.

DECLARATION

```
property TGA_GrayLevel: Boolean;
```

DESCRIPTION

If TGA_GrayLevel is true, the image will be saved as gray levels.

DECLARATION

```
property TGA_ImageName: AnsiString;
```

DESCRIPTION

TGA_ImageName contains the image name of the TGA file.

DECLARATION

```
property TGA_XPos: integer;
```

DESCRIPTION

TGA_XPos is the top-left X coordinate where the image will be shown (not used by ImageEn).

DECLARATION

```
property TGA_YPos: Integer;
```

DESCRIPTION

TGA_YPos is the top-left Y coordinate where the image will be shown (not used by ImageEn).

PXM

DECLARATION

```
property PXM_Comments: TStringList;
```

DESCRIPTION

PXM_Comments contains a list of the comments found inside (or to write into) PPM, PBM or PGM files.

Unit ImageENIO

TIOParamVals

AVI

DECLARATION

```
property AVI_FrameCount: integer;
```

DESCRIPTION

AVI_FrameCount specifies how many frames are contained in the last loaded AVI file.

DECLARATION

```
property AVI_FrameDelayTime: integer;
```

DESCRIPTION

AVI_FrameDelayTime is the time in milliseconds the image will be shown on playing.

IPTC

DECLARATION

```
property IPTC_Info: TIEIPTCInfoList;
```

DESCRIPTION

IPTC_Info contains a list of IPTC information contained in a file. IPTC records can contains text, objects and images. Applications can read/write information from IPTC_Info using string objects or a memory buffer. Each IPTC_Info item has a record number and a dataset number. These values specify the type of data contained in the item, according to IPTC - NAA Information Interchange Model Version 4 (look at www.iptc.org). ImageEn can read/write IPTC textual information of PhotoShop (File info menu). For JPEG files ImageEn read/write IPTC fields from APP13 marker. Click here for a list of IPTC Photoshop items. TIEIPTCInfoList is the object that contains the list of IPTC information.

Example

```
// read the image description (written by
PhotoShop) (caption: var, idx: integer)
ImageEnView1.IO.LoadFromFile('image.jpg');
Idx :=
ImageEnView1.IO.Params.IPTC_Info.IndexOf(2,120);
Caption :=
ImageEnView1.IO.Params.IPTC_Info.StringItem[idx];
// write the image description
ImageEnView1.IO.Params.IPTC_Info.StringItem[idx]:
= 'This is the new caption';
ImageEnView1.IO.SaveToFile('image.jpg');
```

IOParamsVals

Unit ImageENIO
TIOParamVals

Modify the IPTC fields without loading the original image

To load IPTC info from a jpeg, just use LoadFromFile method. After loading the file all IPTC information is in the ImageEnView.IO.Params.IPTC_Info object. To read the caption you can write:

```
ImageEnView.IO.LoadFromFile('image.jpg');
Idx:=ImageEnView.IO.
Params.IPTC_Info.IndexOf(2,120);
Caption:=
ImageEnView.IO.Params.IPTC_Info.StringItem[idx];
//Modify the caption
ImageEnView.IO.Params.IPTC_Info.StringItem[idx]
:= 'new caption';
ImageEnView.IO.SaveToFile('image2.jpg');
```

If you want to modify IPTC info without load the image use ParamsFromFile and InjectJpegIPTC methods, in this way:

```
ImageEnView.IO.ParamsFromFile('one.jpg');
//modify the IPTC info
ImageEnView.IO.InjectJpegIPTC('two.jpg');
```

IOParamsVals

TIEIPTCInfoList

DESCRIPTION

TIEIPTCInfoList contains a list of IPTC information contained in a file. IPTC records can contains text, objects and images. Applications can read/write information from TIEIPTCInfoList using string objects or a memory buffer.

Each IPTC_Info item has a record number and a dataset number. These values specify the type of data contained in the item, according to IPTC - NAA Information Interchange Model Version 4 (look at www.iptc.org).

ImageEn can read/write IPTC textual information of PhotoShop (File info menu).

For JPEG files ImageEn read/write IPTC fields from APP13 marker. Click here for a list of IPTC Photoshop items.

An instance of TIEIPTCInfoList is IPTC_Info

Methods and Properties

DECLARATION

```
function AddBufferItem(RecordNumber: integer;
DataSet: integer; Buffer: pointer; BufferLength:
integer): integer;
```

DESCRIPTION

AddBufferItem adds a memory buffer (non-textual information) with specified RecordNumber and DataSet. AddBufferItem returns the index of the new item, where the first item in the list has an index of 0.

DECLARATION

```
function AddStringItem(RecordNumber: integer;
DataSet: integer; Value: AnsiString): integer;
```

DESCRIPTION

AddStringItem adds a string (textual information) with specified RecordNumber and DataSet. AddStringItem returns the index of the new item, where the first item in the list has an index of 0.

DECLARATION

```
procedure Assign(Source: TIEIPTCInfoList);
```

DESCRIPTION

Assign copies all items from Source.

DECLARATION

```
procedure Clear;
```

DESCRIPTION

Clear removes all items.

DECLARATION

```
property Count: integer;
```

DESCRIPTION

Count returns items count.

DECLARATION

```
property DataSet[idx: integer]:integer;
```

DESCRIPTION

DataSet returns the dataset number associated to the item idx.

DECLARATION

```
procedure DeleteItem(idx: integer);
```

DESCRIPTION

DeleteItem removes the item idx.

DECLARATION

```
function GetItemData(idx: integer): pointer;
```

DESCRIPTION

GetItemData returns the raw data associated to the item idx.

DECLARATION

```
function GetItemLength(idx: integer): integer;
```

DESCRIPTION

GetItemLength returns the raw data length associated to the item idx.

DECLARATION

```
function IndexOf(RecordNumber: integer; DataSet: integer): integer;
```

DESCRIPTION

IndexOf finds the first item that matches with RecordNumber and DataSet parameters and returns its index. If the item is not in the list, IndexOf returns -1.

DECLARATION

```
procedure InsertStringItem(idx: integer; RecordNumber: integer; DataSet: integer; Value: AnsiString);
```

DESCRIPTION

Call InsertStringItem to add item to the middle of the item array. The idx parameter is a zero-based index.

Declaration

```
function LoadFromStandardBuffer(Buffer: pointer;
BufferLength: integer):Boolean;
```

Description

LoadFromStandardBuffer loads IPTC from the specified buffer. LoadFromStandardBuffer is used to extract IPTC info from an image file.

Declaration

```
procedure LoadFromStream(Stream: TStream);
```

Description

LoadFromStream loads all IPTC info from stream. This method cannot load IPTC embedded in image formats.

Declaration

```
property RecordNumber[idx: integer]:integer;
```

Description

Returns the record number associated to the item idx.

DECLARATION

```
procedure SaveToStandardBuffer(var Buffer:
pointer; var BufferLength: integer; WriteHeader:
Boolean);
```

DESCRIPTION

SaveToStandardBuffer saves IPTC in a buffer. You have to free the buffer. This method is used to embed IPTC info inside an image file.

DECLARATION

```
procedure SaveToStream(Stream: TStream);
```

DESCRIPTION

SaveToStream saves IPTC info in the Stream. Do not use this method to embed IPTC info inside image files.

DECLARATION

```
property StringItem[idx: integer]: string;
```

DESCRIPTION

StringItem returns the string associated to the item idx. The item must be of string type.

EXIF tags

DECLARATION

```
property EXIF_ApertureValue: double;
```

DESCRIPTION

EXIF_ApertureValue returns the actual aperture value of the lens when the image was taken. Unit is APEX. To convert this value to ordinary F-number (F-stop), calculate this value's power of root 2 (=1.4142). For example, if the ApertureValue is '5', F-number is 1.41425 = F5.6.

DECLARATION

```
property EXIF_Artist: AnsiString;
```

DESCRIPTION

EXIF_Artist specifies the EXIF artist.

DECLARATION

```
property EXIF_Bitmap: TIEBitmap;
```

DESCRIPTION

Contains thumbnails - not all formats are supported. EXIF_Bitmap is supported only on reading.

DECLARATION

```
property EXIF_BrightnessValue: double;
```

DESCRIPTION

Brightness of taken subject, unit is APEX. To calculate Exposure (Ev) from BrigtnessValue (Bv), you must add SensitivityValue(Sv). Ev=Bv+Sv Sv=log2(ISOSpeedRating/3.125) ISO100:Sv=5, ISO200:Sv=6, ISO400:Sv=7, ISO125:Sv=5.32.

DECLARATION

```
property EXIF_ColorSpace: integer;
```

DESCRIPTION

EXIF_ColorSpace defines Color Space. DCF image must use sRGB color space so value is always '1'. If the picture uses the other color space, value is '65535':Uncalibrated.

DECLARATION

```
property EXIF_CompressedBitsPerPixel: double;
```

DESCRIPTION

The average compression ratio of JPEG (rough estimate)

DECLARATION

```
property EXIF_Contrast: integer;
```

DESCRIPTION

This tag indicates the direction of contrast processing applied by the camera when the image was shot.
0 = Normal
1 = Soft
2 = Hard

DECLARATION

```
property EXIF_Copyright: AnsiString;
```

DESCRIPTION

Shows copyright information.

DECLARATION

```
property EXIF_DateTimeDigitized: AnsiString;
```

DESCRIPTION

Date/Time of image digitized. Usually, it contains the same value of DateTimeOriginal (0x9003). Data format is “YYYY:MM:DD HH:MM:SS”+0x00, total 20 bytes. If clock has not set or digicam doesn’t have clock, the field may be filled with spaces. In the Exif standard, this tag is optional, but it is mandatory for DCF.

DECLARATION

```
property EXIF_DateTimeOriginal: AnsiString;
```

DESCRIPTION

Date/Time that the original image was taken. This value should not be modified by user program. Data format is “YYYY:MM:DD HH:MM:SS”+0x00, total 20bytes. If clock has not set or digicam doesn’t have clock, the field may be filled with spaces. In the Exif standard, this tag is optional, but it is mandatory for DCF.

IOParamsVals

DECLARATION

```
property EXIF_DateTime: AnsiString;
```

DESCRIPTION

EXIF_DateTime returns the Date/Time of image was last modified. Data format is “YYYY:MM:DD HH:MM:SS”+0x00, total 20 bytes. If clock has not set or digicam doesn’t have clock, the field may be filled with spaces. Usually it has the same value of DateTimeOriginal.

DECLARATION

```
property EXIF_DigitalZoomRatio: double;
```

DESCRIPTION

This tag indicates the digital zoom ratio when the image was shot. If the numerator of the recorded value is 0, this indicates that digital zoom was not used.

DECLARATION

```
property EXIF_ExifImageHeight: integer;
```

DESCRIPTION

EXIF_ExifImageHeight is the height of the main image.

DECLARATION

```
property EXIF_ExifImageWidth: integer;
```

DESCRIPTION

EXIF_ExifImageWidth is the width of the main image.

DECLARATION

```
property EXIF_ExifVersion: AnsiString;
```

DESCRIPTION

EXIF_ExifVersion is the Exif version number stored as 4 bytes of ASCII characters. If the picture is based on Exif V2.1, value is “0210”. Since the type is ‘undefined’, there is no NULL (0x00) for termination.

DECLARATION

```
property EXIF_ExposureBiasValue: double;
```

DESCRIPTION

EXIF_ExposureBiasValue is the exposure bias (compensation) value of taking picture. Unit is APEX (EV).

DECLARATION

```
property EXIF_ExposureIndex: double;
```

DESCRIPTION

EXIF_ExposureIndex is the same as EXIF_ISOSpeedRatings (0x8827) but data type is unsigned rational. Only Kodak's digicam uses this tag instead of EXIF_ISOSpeedRatings.

DECLARATION

```
property EXIF_ExposureMode: integer;
```

DESCRIPTION

This tag indicates the exposure mode set when the image was shot. In auto-bracketing mode, the camera shoots a series of frames of the same scene at different exposure settings.
0 = Auto exposure
1 = Manual exposure
2 = Auto bracket

DECLARATION

```
property EXIF_ExposureProgram: integer;
```

DESCRIPTION

EXIF_ExposureProgram is the exposure program that the camera used when image was taken.
1 means manual control
2 program normal
3 aperture priority
4 shutter priority
5 program creative (slow program)
6 program action (high-speed program)
7 portrait mode
8 landscape mode.

DECLARATION

```
property EXIF_ExposureTime: double;
```

DESCRIPTION

EXIF_ExposureTime is the exposure time (reciprocal of shutter speed). Unit is second.

IOParamsVals

DECLARATION

```
property EXIF_FileSource: integer;
```

DESCRIPTION

EXIF_FileSource indicates the image source. Value ‘0x03’ means the image source is a digital still camera.

DECLARATION

```
property EXIF_FlashPixVersion: AnsiString;
```

DESCRIPTION

EXIF_FlashPixVersion stores FlashPix version. If the image data is based on FlashPix format Ver.1.0, value is “0100”.

IOParamsVals

DECLARATION

```
property EXIF_Flash: Integer;
```

DESCRIPTION

0 means flash did not fire
1 flash fired
5 flash fired but strobe return light not detected
7 flash fired and strobe return light detected

DECLARATION

```
property EXIF_FNumber: double;
```

DESCRIPTION

EXIF_FNumber is the actual F-number (F-stop) of lens when the image was taken.

DECLARATION

```
property EXIF_FocalLength: double;
```

DESCRIPTION

Focal length of lens used to take image. Unit is millimeters.

DECLARATION

```
property EXIF_FocalLengthIn35mmFilm: integer;
```

DESCRIPTION

This tag indicates the equivalent focal length assuming a 35mm film camera, in mm. A value of 0 means the focal length is unknown. Note that this tag differs from the FocalLength tag.

DECLARATION

```
property EXIF_FocalPlaneResolutionUnit:integer;
```

DESCRIPTION

Unit of FocalPlaneXResoluton/FocalPlaneYResolution:
1 means no-unit
2 inch
3 centimeter
Note: Some of Fujifilm's digicams (e.g. FX2700, FX2900, Finepix 4700Z/40i, etc.) use value '3' so it must be 'centimeter', but it seems that they use a '8.3mm?' (1/3in.?) to their ResolutionUnit. Fuji's BUG? Finepix4900Z has been changed to use value '2' but it doesn't match to actual value either.

DECLARATION

```
property EXIF_FocalPlaneXResolution: double;
```

DESCRIPTION

EXIF_FocalPlaneXResolution is the pixel density at CCD's position. If you have Megapixel digicam and take a picture by lower resolution (e.g.VGA mode), this value is re-sampled by picture resolution. In such case, FocalPlaneResolution is not same as CCD's actual resolution.

DECLARATION

```
property EXIF_FocalPlaneYResolution: double;
```

DESCRIPTION

EXIF_FocalPlaneYResolution is the pixel density at CCD's position. If you have Megapixel digicam and take a picture by lower resolution (e.g. VGA mode), this value is re-sampled by picture resolution. In such case, FocalPlaneResolution is not same as CCD's actual resolution.

DECLARATION

```
property EXIF_GainControl: integer;
```

DESCRIPTION

This tag indicates the degree of overall image gain adjustment.
0 = None
1 = Low gain up
2 = High gain up
3 = Low gain down
4 = High gain down

DECLARATION

```
property EXIF_HasEXIFData: Boolean;
```

DESCRIPTION

If EXIF_HasEXIFData is true, the loaded image contains EXIF information tags. If you want to maintain the original EXIF info, set EXIF_HasEXIFData to False, before saving.

Important Note: If an image is assigned from a TImageEnView to another, the EXIF tags is NOT automatically assigned. To maintain EXIF data in the second TImageEnView or TImageEnVect, you must also assign the EXIF Data: This applies to all ImageEn 'Display' components including TImageEnView and TImageEnVect.

Here is typical situations:
1) You want to maintain EXIF untouched:
nothing to do...
2) you want change some EXIF fields, for example:
Params.EXIF_Software:='ImageEn';
Params.EXIF_HasEXIFData:=true;
3) you wnat remove all EXIFs:
Params.ResetInfo;

IOParamsVals

DECLARATION

```
property EXIF_ImageDescription: AnsiString;
```

DESCRIPTION

EXIF_ImageDescription describes the image.

DECLARATION

```
property EXIF_ImageUniqueID: AnsiString;
```

DESCRIPTION

This tag indicates an identifier assigned uniquely to each image. It is recorded as an ASCII string equivalent to hexadecimal notation and 128-bit fixed length.

DECLARATION

```
property EXIF_ISOSpeedRatings[index: integer]:
integer;
```

DESCRIPTION

EXIF_ISOSpeedRatings is the CCD sensitivity equivalent to Ag-Hr film speedrate Index is in the range 0..1. ”0” for all values means “unspecified”.

DECLARATION

```
property EXIF_LightSource: integer;
```

DESCRIPTION

Light source, actually this means white balance setting:
0 means unknown
1 daylight
2 fluorescent
3 tungsten
10 flash
17 standard light A
18 standard light B
19 standard light C
20 D55
21 D65
22 D75
255 other

DECLARATION

```
property EXIF_Make: AnsiString;
```

DESCRIPTION

EXIF_Make is the manufacturer of the recording equipment. This is the manufacturer of the DSC, scanner, video digitizer or other equipment that generated the image.

DECLARATION

```
property EXIF_MakerNote: TIETagsHandler;
```

DESCRIPTION

EXIF_MakerNote contains maker specific information.
Unfortunately there is a standard for this tag, then we a general handler which read the IFD (when present) of the maker note.
We suggest looking at inputoutput/EXIF demo for more details.

DECLARATION

```
property EXIF_MaxApertureValue: double;
```

DESCRIPTION

EXIF_MaxApertureValue is the maximum aperture value of the lens. Convert to F-number by calculating power of root 2 (same process as ApertureValue).

DECLARATION

```
property EXIF_MeteringMode: integer;
```

DESCRIPTION

Exposure metering method:
0 means unknown
1 average
2 center weighted average
3 spot
4 multi-spot
5 multi-segment
6 partial
255 other

DECLARATION

```
property EXIF_Model: AnsiString;
```

DESCRIPTION

Shows model number of digicam

Unit ImageENIO
TIOParamVals

DECLARATION

```
property EXIF_Orientation: integer;
```

DESCRIPTION

The orientation of the camera relative to the scene, when the image was captured

Value	Description
1	top left side
2	top right side
3	bottom right side
4	bottom left side
5	left side top
6	right side top
7	right side bottom
8	left side bottom

DECLARATION

```
property EXIF_PrimaryChromaticities[index:
integer]: double;
```

DESCRIPTION

EXIF_PrimaryChromaticities defines chromaticity of the primaries of the image. If the image uses CCIR Recommendation 709 primaries, the values are
‘640/1000,330/1000,300/1000,600/1000,150/1000,0/1000’.
index is in the range 0..5.
”-1” for all values means “unspecified”.

DECLARATION

```
property EXIF_ReferenceBlackWhite[index:
integer]: double;
```

DESCRIPTION

Shows reference value of black point/white point. In case of YCbCr format, first 2 show black/white of Y, next 2 are Cb, last 2 are Cr. In case of RGB format, first 2 show black/white of R, next 2 are G, last 2 are B.
”-1” for all values means “unspecified”.
index is in the range 0..5.

DECLARATION

```
property EXIF_RelatedSoundFile: AnsiString;
```

DESCRIPTION

If this digicam can record audio data with image, shows name of audio data.

DECLARATION

```
property EXIF_ResolutionUnit: integer;
```

DESCRIPTION

EXIF_ResolutionUnit is the Unit of XResolution/YResolution.
1 means no unit
2 means inch
3 means centimeter
Default value is 2 (inch)

DECLARATION

```
property EXIF_Saturation: integer;
```

DESCRIPTION

This tag indicates the direction of saturation processing applied by the camera when the image was shot.
0 = Normal
1 = Low saturation
2 = High saturation

DECLARATION

```
property EXIF_SceneCaptureType: integer;
```

DESCRIPTION

This tag indicates the type of scene that was shot. It can also be used to record the mode in which the image was shot. Note that this differs from the scene type (SceneType) tag.
0 = Standard
1 = Landscape
2 = Portrait
3 = Night scene

DECLARATION

```
property EXIF_SceneType: integer;
```

DESCRIPTION

EXIF_SceneType indicates the type of scene. Value of ‘0x01’ means that the image was directly photographed.

DECLARATION

```
property EXIF_SensingMethod: integer;
```

DESCRIPTION

EXIF_SensingMethod shows type of image sensor unit. ‘2’ means 1 chip color area sensor. Most all digicams use this type.

DECLARATION

```
property EXIF_Sharpness: integer;
```

DESCRIPTION

This tag indicates the direction of sharpness processing applied by the camera when the image was shot.
0 = Normal
1 = Soft
2 = Hard

DECLARATION

```
property EXIF_ShutterSpeedValue: double;
```

DESCRIPTION

EXIF_ShutterSpeedValue is the shutter speed by APEX value. To convert this value to ordinary 'Shutter Speed'; calculate this value's power of 2, then reciprocal. For example, if the ShutterSpeedValue is '4', shutter speed is 1/(24)=1/16 second.

DECLARATION

```
property EXIF_Software: AnsiString;
```

DESCRIPTION

EXIF_Software shows firmware(internal software of digicam) version number

DECLARATION

```
property EXIF_SubjectDistanceRange: integer;
```

DESCRIPTION

This tag indicates the distance to the subject.
0 = unknown
1 = Macro
2 = Close view
3 = Distant view

DECLARATION

```
property EXIF_SubjectDistance: double;
```

DESCRIPTION

Distance to focus point, unit is meters.

DECLARATION

```
property EXIF_SubsecTimeDigitized: AnsiString;
```

DESCRIPTION

Some of digicam can take 2~30 pictures per second, but DateTime/DateTimeOriginal/DateTimeDigitized tag can't record the sub-second time. SubsecTime tag is used to record it.
For example, DateTimeOriginal = "1996:09:01 09:15:30", SubSecTimeOriginal = "130", Combined original time is "1996:09:01 09:15:30.130"

IOParamsVals

DECLARATION

```
property EXIF_SubsecTimeOriginal: AnsiString;
```

DESCRIPTION

Some of digicam can take 2~30 pictures per second, but DateTime/DateTimeOriginal/DateTimeDigitized tag can't record the sub-second time. SubsecTime tag is used to record it.
For example, DateTimeOriginal = "1996:09:01 09:15:30", SubSecTimeOriginal = "130", Combined original time is "1996:09:01 09:15:30.130"

DECLARATION

```
property EXIF_SubsecTime: AnsiString;
```

DESCRIPTION

Some of digicam can take 2~30 pictures per second, but DateTime/DateTimeOriginal/DateTimeDigitized tag can't record the sub-second time. SubsecTime tag is used to record it.
For example, DateTimeOriginal = "1996:09:01 09:15:30", SubSecTimeOriginal = "130", Combined original time is "1996:09:01 09:15:30.130"

Unit ImageENIO
TIOParamVals

DECLARATION

```
property EXIF_UserCommentCode: AnsiString;
```

DESCRIPTION

EXIF_UserCommentCode specifies the character code (8 bytes) for the EXIF_UserComment property. Allowed values:

Character Code	Code Designation (8Bytes)	References
ASCII	41H, 53.H, 43.H, 49.H, 49H, 00.H, 00.H, 00.H	ITU-T T.50 IA5
JIS	4A.H, 49.H, 53.H, 00.H, 00.H, 00.H, 00.H, 00.H	JIS X0208-1990
Unicode	55.H, 4EH, 49.H, 43.H, 4F.H, 44.H, 45.H, 00.H	Unicode Standard
Undefined	00.H, 00.H, 00.H, 00.H, 00.H, 00.H, 00.H, 00.H	Undefined

DECLARATION

```
property EXIF_UserComment: WideString;
```

DESCRIPTION

EXIF_UserComment is a tag to write keywords or comments on the image besides those in EXIF_ImageDescription, and without the character code limitations of the EXIF_ImageDescription tag. The character code is specified by the EXIF_UserCommentCode property.

It is necessary to specify how the string will be coded (ASCII or Unicode) when using the EXIF_UserCommentCode.

Example:

```
ImageEnView1.IO.LoadFromFile('input.jpg');
ImageEnView1.IO.Params.EXIF_UserComment := 'Hello
World!';
ImageEnView1.IO.Params.EXIF_UserCommentCode :=
IEEXIFUSERCOMMENTCODE_UNICODE;
ImageEnView1.IO.Params.EXIF_HasEXIFData := true;
ImageEnView1.IO.SaveToFileTIFF('test.tiff');
```

...and read back....

```
ImageEnView1.IO.LoadFromFileTIFF('test.tiff');
ShowMessage( ImageEnView1.IO.Params.EXIF_UserComment
);
```

IEEXIFUSERCOMMENTCODE_UNICODE is defined in the ImageEnIO unit.

DECLARATION

```
property EXIF_WhiteBalance: integer;
```

DESCRIPTION

This tag indicates the white balance mode set when the image was shot.
0 = Auto white balance
1 = Manual white balance

DECLARATION

```
property EXIF_WhitePoint[index: integer]: double;
```

DESCRIPTION

EXIF_WhitePoint defines chromaticity of white points of the image. If the image uses CIE Standard Illumination D65 (known as international standard of 'daylight'), the values are '3127/10000, 3290/10000'.
index is in the range 0..1. "-1" for both values means "unspecified".

DECLARATION

```
property EXIF_XResolution: double;
```

DESCRIPTION

EXIF_XResolution is the resolution of image. Default value is 1/72 inch (one 'point'), but it has no meaning because personal computers typically don't use this value to display/print.

DECLARATION

```
property EXIF_YCbCrCoefficients[index:
integer]:double;
```

DESCRIPTION

When the image format is YCbCr, this value shows a constant to translate it to RGB format. Usually values are '0.299/0.587/0.114'. index is in the range 0..2. "-1" for all values means "unspecified".

DECLARATION

```
property EXIF_YCbCrPositioning: integer;
```

DESCRIPTION

When image format is YCbCr and uses ‘Subsampling’ (cropping of chroma data, all digicams do that), defines the chroma sample point of subsampling pixel array.
0 means “unspecified”
1 means the center of pixel array.
2 means the datum point.

DECLARATION

```
property EXIF_YResolution: double;
```

DESCRIPTION

EXIF_YResolution is the resolution of image. Default value is 1/72 inch (one ‘point’), but it has no meaning because personal computers typically don’t use this value to display/print.

EXIF Windows XP tags

Declaration

```
property EXIF_XPAuthor: WideString;
```

Description

EXIF_XPAuthor specifies Windows XP image author. This is shown in properties of jpeg/tiff file under Windows XP.

Declaration

```
property EXIF_XPComment: WideString;
```

Description

EXIF_XPComment specifies Windows XP image comment. This is shown in properties of jpeg/tiff file under Windows XP.

Declaration

```
property EXIF_XPKeywords: WideString;
```

Description

EXIF_XPKeywords specifies Windows XP image keywords. This is shown in properties of jpeg/tiff file under Windows XP.

Declaration

```
property EXIF_XPRating: integer;
```

Description

EXIF_XPRating specifies Windows XP image rating. This is shown in properties of jpeg/tiff file under Windows XP. Allowed values are from 0 up to 5. -1 means not available.

Declaration

```
property EXIF_XPSubject: WideString;
```

Description

EXIF_XPSubject specifies Windows XP image subject. This is shown in properties of jpeg/tiff file under Windows XP.

Declaration

```
property EXIF_XPTitle: WideString;
```

Description

EXIF_XPTitle specifies Windows XP image title. This is shown in properties of jpeg/tiff file under Windows XP.

EXIF GPS tags

Declaration

```
property EXIF_GPSVersionID: AnsiString;
```

Description

EXIF_GPSVersionID indicates the version of GPSInfoIFD.

Declaration

```
property EXIF_GPSLatitudeRef: AnsiString;
```

Description

EXIF_GPSLatitudeRef indicates whether the latitude is north or south latitude. The ASCII value ‘N’ indicates north latitude, and ‘S’ is south latitude.

Declaration

```
property EXIF_GPSLatitudeDegrees: double;
```

Description

EXIF_GPSLatitudeDegrees indicates the latitude degrees.

DECLARATION

```
property EXIF_GPSLatitudeMinutes: double;
```

DESCRIPTION

EXIF_GPSLatitudeMinutes indicates the latitude minutes.

DECLARATION

```
property EXIF_GPSLatitudeSeconds: double;
```

DESCRIPTION

EXIF_GPSLatitudeSeconds indicates the latitude seconds.

DECLARATION

```
property EXIF_GPSLongitudeRef: AnsiString;
```

DESCRIPTION

EXIF_GPSLongitudeRef indicates whether the longitude is east or west longitude. ASCII ‘E’ indicates east longitude, and ‘W’ is west longitude.

DECLARATION

```
property EXIF_GPSLongitudeDegrees: double;
```

DESCRIPTION

EXIF_GPSLongitudeDegrees indicates the longitude degrees.

DECLARATION

```
property EXIF_GPSLongitudeMinutes: double;
```

DESCRIPTION

EXIF_GPSLongitudeMinutes indicates the longitude minutes.

DECLARATION

```
property EXIF_GPSLongitudeSeconds: double;
```

DESCRIPTION

EXIF_GPSLongitudeSeconds indicates the longitude seconds.

DECLARATION

```
property EXIF_GPSAltitudeRef: AnsiString;
```

DESCRIPTION

EXIF_GPSAltitudeRef indicates the altitude used as the reference altitude. If the reference is sea level and the altitude is above sea level, '0' is given. If the altitude is below sea level, a value of '1' is given and the altitude is indicated as an absolute value in the EXIF_GPSAltitude tag. The reference unit is meters.

DECLARATION

```
property EXIF_GPSAltitude: double;
```

DESCRIPTION

EXIF_GPSAltitude indicates the altitude based on the reference in GPSAltitudeRef. The reference unit is meters.

DECLARATION

```
property EXIF_GPSTimeStampHour: double;
```

DESCRIPTION

EXIF_GPSTimeStampHour indicates the time as UTC (Coordinated Universal Time).

DECLARATION

```
property EXIF_GPSTimeStampMinute: double;
```

DESCRIPTION

EXIF_GPSTimeStampMinute indicates the time as UTC (Coordinated Universal Time).

DECLARATION

```
property EXIF_GPSTimeStampSecond: double;
```

DESCRIPTION

EXIF_GPSTimeStampSecond indicates the time as UTC (Coordinated Universal Time).

DECLARATION

```
property EXIF_GPSSatellites: AnsiString
```

DESCRIPTION

Indicates the GPS satellites used for measurements. This tag can be used to describe the number of satellites, their ID number, angle of elevation, azimuth, SNR and other information in ASCII notation. The format is not specified.

DECLARATION

```
property EXIF_GPSStatus: AnsiString;
```

DESCRIPTION

EXIF_GPSStatus indicates the status of the GPS receiver when the image is recorded. 'A' means measurement is in progress, and 'V' means the measurement is Interoperability.

DECLARATION

```
property EXIF_GPSMeasureMode: AnsiString;
```

DESCRIPTION

EXIF_GPSMeasureMode indicates the GPS measurement mode. '2' means two-dimensional measurement and '3' means three-dimensional measurement is in progress.

DECLARATION

```
property EXIF_GPSDOP: double;
```

DESCRIPTION

EXIF_GPSDOP indicates the GPS DOP (data degree of precision). An HDOP value is written during two-dimensional measurement and PDOP during three-dimensional measurement.

DECLARATION

```
property EXIF_GPSSpeedRef: AnsiString;
```

DESCRIPTION

EXIF_GPSSpeedRef indicates the unit used to express the GPS receiver speed of movement. ‘K’ ‘M’ and ‘N’ represents kilometers per hour, miles per hour, and knots.

DECLARATION

```
property EXIF_GPSSpeed: double;
```

DESCRIPTION

EXIF_GPSSpeed indicates the speed of GPS receiver movement.

DECLARATION

```
property EXIF_GPSTrackRef: AnsiString;
```

DESCRIPTION

EXIF_GPSSpeed indicates the reference for giving the direction of GPS receiver movement. ‘T’ denotes true direction and ‘M’ is magnetic direction.

DECLARATION

```
property EXIF_GPSTrack: double;
```

DESCRIPTION

EXIF_GPSTrack indicates the direction of GPS receiver movement. The range of values is from 0.00 to 359.99.

DECLARATION

```
property EXIF_GPSImgDirectionRef: AnsiString;
```

DESCRIPTION

EXIF_GPSImgDirectionRef indicates the reference for giving the direction of the image when it is captured. ‘T’ denotes true direction and ‘M’ is magnetic direction.

DECLARATION

```
property EXIF_GPSImgDirection: double;
```

DESCRIPTION

EXIF_GPSImgDirection indicates the direction of the image when it was captured. The range of values is from 0.00 to 359.99.

Declaration

```
property EXIF_GPSMapDatum: AnsiString;
```

Description

EXIF_GPSMapDatum indicates the geodetic survey data used by the GPS receiver. If the survey data is restricted to Japan, the value of this tag is 'TOKYO' or 'WGS-84'.

Declaration

```
property EXIF_GPSDestLatitudeRef: AnsiString;
```

Description

EXIF_GPSDestLatitudeRef indicates whether the latitude of the destination point is north or south latitude. The ASCII value 'N' indicates north latitude, and 'S' is south latitude.

Declaration

```
property EXIF_GPSDestLatitudeDegrees: double;
```

Description

Indicates the latitude degrees of the destination point

DECLARATION

```
property EXIF_GPSDestLatitudeMinutes: double;
```

DESCRIPTION

EXIF_GPSDestLatitudeMinutes indicates the latitude minutes of the destination point.

DECLARATION

```
property EXIF_GPSDestLatitudeSeconds: double;
```

DESCRIPTION

EXIF_GPSDestLatitudeSeconds indicates the latitude seconds of the destination point.

DECLARATION

```
property EXIF_GPSDestLongitudeRef: AnsiString;
```

DESCRIPTION

EXIF_GPSDestLongitudeRef indicates whether the longitude of the destination point is east or west longitude. ASCII ‘E’ indicates east longitude, and ‘W’ is west longitude.

DECLARATION

```
property EXIF_GPSDestLongitudeDegrees: double;
```

DESCRIPTION

EXIF_GPSDestLongitudeDegrees indicates the longitude degrees of the destination point.

DECLARATION

```
property EXIF_GPSDestLongitudeMinutes: double;
```

DESCRIPTION

IEXIF_GPSDestLongitudeMinutes indicates the longitude minutes of the destination point.

DECLARATION

```
property EXIF_GPSDestLongitudeSeconds: double;
```

DESCRIPTION

EXIF_GPSDestLongitudeSeconds indicates the longitude seconds of the destination point.

DECLARATION

```
property EXIF_GPSDestBearingRef: AnsiString;
```

DESCRIPTION

EXIF_GPSDestBearingRef indicates the reference used for giving the bearing to the destination point. 'T' denotes true direction and 'M' is magnetic direction.

DECLARATION

```
property EXIF_GPSDestBearing: double;
```

DESCRIPTION

EXIF_GPSDestBearing indicates the bearing to the destination point. The range of values is from 0.00 to 359.99.

DECLARATION

```
property EXIF_GPSDestDistanceRef: AnsiString;
```

DESCRIPTION

EXIF_GPSDestDistanceRef indicates the unit used to express the distance to the destination point. 'K', 'M' and 'N' represent kilometers, miles and knots.

DECLARATION

```
property EXIF_GPSDestDistance: double;
```

DESCRIPTION

EXIF_GPSDestDistance is the distance to destination.

DECLARATION

```
property EXIF_GPSDateStamp: AnsiString;
```

DESCRIPTION

GPS date.

EXIF Helper Functions

The following functions are EXIF helper functions developed by the author. These functions convert EXIF integer values to a string.

DECLARATION

```
function EXIF_ExposureProgramToStr (
EXIF_ExposureProgram: integer ): string;
// Convert EXIF_ExposureProgram to a string
begin
  case EXIF_ExposureProgram of
    0: Result := 'Unknown';
    1: Result := 'Manual control';
    2: Result := 'Normal';
    3: Result := 'Aperture priority';
    4: Result := 'Shutter priority';
    5: Result := 'Creative (slow program)';
    6: Result := 'Action(high-speed program)';
    7: Result := 'Portrait mode';
    8: Result := 'Landscape mode'
  else
    Result := 'Unknown';
  end; // case
end;
```

DECLARATION

```
function EXIF_LightSourceToStr (
EXIF_LightSource: integer ): string;
// Convert EXIF_LightSource to a string
begin
  case EXIF_LightSource of
    0: Result := 'Unknown';
    1: Result := 'Daylight';
    2: Result := 'Fluorescent';
    3: Result := 'Tungsten';
    10: Result := 'Flash';
    17: Result := 'Standard light A';
    18: Result := 'Standard light B';
    19: Result := 'Standard light C';
    20: Result := 'D55';
    21: Result := 'D65';
    22: Result := 'D75';
    255: Result := 'Other'
  else
    Result := 'Unknown';
  end;
end; // case
```

DECLARATION

```
function EXIF_FlashToStr ( EXIF_Flash: integer ): string;
// Convert EXIF_Flash value to a string
begin
  case EXIF_Flash of
    0: Result := 'Unknown';
    1: Result := 'Flash fired';
    5: Result := 'Flash fired but strobe return light not detected';
    7: Result := 'Flash fired and strobe return light detected';
    9: Result := 'Flash fired, compulsory flash mode';
    25: Result := '(25) Onboard Flash fired';
    73: Result := '(73) Onboard Flash fired';
    89: Result := '(89) Onboard Flash fired'
  else
    Result := 'Unknown';
  end;
end; // case
```

DECLARATION

```
function EXIF_MeteringModeToString (
EXIF_MeteringMode: integer ): string;
// Convert EXIF_MeteringMode to string
begin
  case EXIF_MeteringMode of
    0: Result := 'Unknown';
    1: Result := 'Average';
    2: Result := 'Center weighted average';
    3: Result := 'Spot';
    4: Result := 'Multi-spot';
    5: Result := 'Multi-segment';
    6: Result := 'Partial';
    255: Result := 'Other'
  else
    Result := 'Unknown';
  end; // case;
end;
```

DECLARATION

```
function EXIF_SensingMethodToString (
EXIF_SensingMethod: integer ): string;
// Convert EXIF_SensingMethod value to a string
begin
  case EXIF_SensingMethod of
    0: Result := 'Unknown';
    2: Result := '1 chip color area sensor';
  else
    Result := 'Unknown';
  end;
end; // case
```

DECLARATION

```
function EXIF_ColorSpaceToString (
EXIF_ColorSpace: integer ): string;
// EXIF_ColorSpace
begin
  case EXIF_ColorSpace of
    0: Result := 'Inknown';
    1: Result := 'sRGB';
    65535: Result := 'Uncalibrated'
  else
    Result := 'Inknown';
  end;
end; // case
```

IOParamsVals

DECLARATION

```
function EXIF_FocalPlaneResolutionUnitToString (
EXIF_FocalPlaneResolutionUnit: integer ): string;
begin
  // EXIF_FocalPlaneResolutionUnit
  case EXIF_FocalPlaneResolutionUnit of
    0: Result := 'Unknown';
    1: Result := 'No-unit';
    2: Result := 'Inches';
    3: Result := 'Centimeter'
  else
    Result := 'Unknown';
  end; // case
end;
```

DECLARATION

```
function EXIF_ResolutionUnitToString (
EXIF_ResolutionUnit: integer ): string;
// Convert EXIF_ResolutionUnit value to a string
begin
  case EXIF_ResolutionUnit of
    0: Result := 'Unknown';
    1: Result := 'No-unit';
    2: Result := 'Inches';
    3: Result := 'Centimeter';
  end; // case
end;
```

DECLARATION

```
function Exif_DateToDateTime ( dstr: string ):
TDateTime;
// Convert ExifDate To DateTime
type
  TConvert = packed record
    year: array [ 1..4 ] of Char;
    f1: Char;
    mon: array [ 1..2 ] of Char;
    f2: Char;
    day: array [ 1..2 ] of Char;
    f3: Char;
    hr: array [ 1..2 ] of Char;
    f4: Char;
    min: array [ 1..2 ] of Char;
    f5: Char;
    sec: array [ 1..2 ] of Char;
  end;
```

DECLARATION

```
function EXIF_YCbCrPositioningToString (
EXIF_YCbCrPositioning: integer ): string;
// Convert EXIF_YCbCrPositioning value to a string
begin
  case EXIF_YCbCrPositioning of
    0: Result := 'Unknown';
    1: Result := 'Center of pixel array';
    2: Result := 'Datum point'
  else
    Result := 'Unknown';
  end; // case
end;
```

IOParamsVals

DECLARATION

```
function EXIF_OrientationToString (
EXIF_Orientation: integer ): string;
// Convert EXIF_Orientation value to a string
begin
  case EXIF_Orientation of
    0: Result := 'Unknown';
    1: Result := 'Top left side';
    2: Result := 'Top right side';
    3: Result := 'Bottom right side';
    4: Result := 'Bottom left side';
    5: Result := 'Left side top';
    6: Result := 'Right side top';
    7: Result := 'Right side bottom';
    8: Result := 'Left side bottom'
  else
    Result := 'Unknown';
  end; // case
end;
```

DECLARATION

```
function HasIPTCInfo( const AImageEnView:
TImageEnView ): boolean;
// Return True if AImageEnView has IPTC Info
var
  k: integer;
begin
   with AImageEnView.IO.Params.IPTC_Info do
      k := IndexOf( iptc[ 0 ].R, iptc[ 0 ].d );
   Result := k >= 0;
end;
```

IOParamsVals

EXIF GPS Helper Functions

By Nigel Cross

Declaration

```
function DegreesMinutesSecondsToDecimal(dDeg,
dMin, dSec: Double): Extended;
begin
  Result := dDeg + (dMin / 60) + (dSec / 3600);
end;
```

Declaration

```
function GPSAsDecimal(const dGPSDegrees,
dGPSMinutes, dGPSSeconds: Double; const
sGPSReference: string): Extended;
begin
  Result :=
DegreesMinutesSecondsToDecimal(dGPSDegrees,
dGPSMinutes, dGPSSeconds);
  if SameText(RemoveNull(sGPSReference), 'S') or
     SameText(RemoveNull(sGPSReference), 'W')
then
    Result := -1 * Result;
end;
```

DECLARATION

```
function GPSLatitudeDecimal(AParams:
TIOParamsVals): Extended;
begin
  with AParams do
    Result :=
      GPSAsDecimal(EXIF_GPSLatitudeDegrees,
      EXIF_GPSLatitudeMinutes,
      EXIF_GPSLatitudeSeconds,
      EXIF_GPSLatitudeRef);
end;
```

DECLARATION

```
function GPSLongitudeDecimal(AParams:
TIOParamsVals): Extended;
begin
  with AParams do
    Result :=
    GPSAsDecimal(EXIF_GPSLongitudeDegrees,
    EXIF_GPSLongitudeMinutes,
    EXIF_GPSLongitudeSeconds,
    EXIF_GPSLongitudeRef);
end;
```

IOParamsVals

Unit ImageENIO

TIOParamVals

DECLARATION

```
function HasExifGPSData(AParams: TIOParamsVals):
Extended;
begin
  Result := (GPSLatitudeDecimal(AParams) <> 0) or
            (GPSLongitudeDecimal(AParams) <> 0)
end;
```

IOParamsVals

DECLARATION

```
// return the GPS fields from the exif fields
function ExifGPSData(sFilename: string;
           AnImageEnIO: TImageEnIO;
           out xGPSLatitudeDegrees: Extended;
           out xGPSLongitudeDegrees: Extended
): Boolean;

begin
  result := False;
  xGPSLatitudeDegrees  := 0;
  xGPSLongitudeDegrees := 0;
  if IsJPEG(sFilename, TRUE) then
  try
      // use get params from file to fill the
      // _ImageEnIO with the EXIF data
    AnImageEnIO.paramsfromfile(sFilename);
    if AnImageEnIO.params.EXIF_HasEXIFData then
    begin
      xGPSLatitudeDegrees  :=
        GPSLatitudeDecimal(AnImageEnIO.Params);
      xGPSLongitudeDegrees :=
        GPSLongitudeDecimal(AnImageEnIO.Params);
end;
    Result := (xGPSLatitudeDegrees <> 0) or
              (xGPSLongitudeDegrees <> 0);
 except
    // ERROR
  end;
end;
```

PostScript

Declaration

```
property PS_Compression: TIOPSCompression;
```

Description

PS_Compression specifies the compression filter for a PostScript file.

Example

```
ImageEnView1.IO.LoadFromFile('input.tif');
ImageEnView1.IO.Params.PS_Compression :=
ioPS_G4FAX;
ImageEnView1.IO.SaveToFile('output.ps');
```

Unit ImageENIO
TIOParamVals

DECLARATION

```
property PS_PaperHeight: integer
```

DESCRIPTION

PS_PaperHeight specifies the paper height in PostScript points (1 point=1/72 of inch). Default values are width=595 and height=842 (A4 format).
Common values are:

Paper Size	PS_PaperWidth	PS_PaperHeight
A0	2380	3368
A1	1684	2380
A2	1190	1684
A3	842	1190
A4	595	842
A5	421	595
A6	297	421
B5	501	709
Letter (8.5”x11”)	612	792
Legal (8.5”x14”)	612	1008
Ledger (17”x11”)	1224	792
11”x17”	792	1224

Example

```
// save using letter paper size
ImageEnView1.IO.Params.PS_PaperWidth := 612;
ImageEnView1.IO.Params.PS_PaperHeight := 792;
ImageEnView1.IO.SaveToFile('output.ps');
```

Unit ImageENIO
TIOParamVals

DECLARATION

```
property PS_PaperWidth: integer
```

DESCRIPTION

PS_PaperWidth specifies the paper width in PostScript points (1 point=1/72 of inch). Default values are width=595 and height=842 (A4 format).
Common values are:

Paper Size	PS_PaperWidth	PS_PaperHeight
A0	2380	3368
A1	1684	2380
A2	1190	1684
A3	842	1190
A4	595	842
A5	421	595
A6	297	421
B5	501	709
Letter (8.5”x11”)	612	792
Legal (8.5”x14”)	612	1008
Ledger (17”x11”)	1224	792
11”x17”	792	1224

Example

```
// save using letter paper size
ImageEnView1.IO.Params.PS_PaperWidth := 612;
ImageEnView1.IO.Params.PS_PaperHeight := 792;
ImageEnView1.IO.SaveToFile('output.ps');
```

DECLARATION

```
property PS_Title: AnsiString;
```

DESCRIPTION

PS_Title specifies the title of the PostScript file.

Adobe PDF

DECLARATION

```
property PDF_Author: AnsiString;
```

DESCRIPTION

PDF_Author specifies the name of the person who created the document.

DECLARATION

```
property PDF_Compression: TIOPDFCompression;
```

DESCRIPTION

PDF_Compression specifies the compression filter for an Adobe PDF file.

DECLARATION

```
property PDF_Creator: AnsiString;
```

DESCRIPTION

PDF_Creator specifies the name of the application that created the original document.

DECLARATION

```
property PDF_Keywords: AnsiString;
```

DESCRIPTION

PDF_Keywords specifies the keywords associated with the document.

Unit ImageENIO
TIOParamVals

DECLARATION

```
property PDF_PaperHeight: integer
```

DESCRIPTION

PDF_PaperHeight specifies the paper height in Adobe PDF points (1 point=1/72 of inch). Default values are width=595 and height=842 (A4 format).
Common values are:

Paper Size	PDF_PaperWidth	PDF_PaperHeight
A0	2380	3368
A1	1684	2380
A2	1190	1684
A3	842	1190
A4	595	842
A5	421	595
A6	297	421
B5	501	709
Letter (8.5”x11”)	612	792
Legal (8.5”x14”)	612	1008
Ledger (17”x11”)	1224	792
11”x17”	792	1224

Unit ImageENIO
TIOParamVals

DECLARATION

```
property PDF_PaperWidth: integer
```

DESCRIPTION

PDF_PaperWidth specifies the paper width in Adobe PDF points (1 point=1/72 of inch). Default values are width=595 and height=842 (A4 format).
Common values are:

Paper Size	PDF_PaperWidth	PDF_PaperHeight
A0	2380	3368
A1	1684	2380
A2	1190	1684
A3	842	1190
A4	595	842
A5	421	595
A6	297	421
B5	501	709
Letter (8.5”x11”)	612	792
Legal (8.5”x14”)	612	1008
Ledger (17”x11”)	1224	792
11”x17”	792	1224

DECLARATION

```
property PDF_Producer: AnsiString;
```

DESCRIPTION

PDF_Producer specifies the name of the application that converted the image to PDF.

DECLARATION

```
property PDF_Subject: AnsiString;
```

DESCRIPTION

PDF_Subject specifies the subject of the document.

DECLARATION

```
property PDF_Title: AnsiString;
```

DESCRIPTION

PDF_Title specifies the PDF document's title.

RAW (Camera)

DECLARATION

```
property RAW_AutoAdjustColors: Boolean;
```

DESCRIPTION

If RAW_AutoAdjustColors is true ImageEn applies an algorithm to adjust image colors. This works only when the RAW file contains EXIF thumbnails with correct colors, because ImageEn gets right colors from the thumbnail.

Example

```
ImageEnView1.IO.Params.RAW_AutoAdjustColors :=
True;
ImageEnView1.IO.LoadFromFile('input.crw');
```

DECLARATION

```
property RAW_BlueScale: double
```

DESCRIPTION

This property specifies the blue multiplier used when loading the RAW. The default is 1.0 (daylight).

Example

```
ImageEnView1.IO.Params.RAW_BlueScale := 0.8;
ImageEnView1.IO.LoadFromFile('CRW_0001.CRW');
```

DECLARATION

```
property RAW_Bright: double
```

DESCRIPTION

This property specifies the brightness value used when loading the RAW. The default is 1.0.

Example

```
ImageEnView1.IO.Params.RAW_Bright := 1.5;
ImageEnView1.IO.LoadFromFile('CRW_0001.CRW');
```

DECLARATION

```
property RAW_Camera: AnsiString;
```

DESCRIPTION

RAW_Camera returns the camera descriptor.

Example

```
ImageEnView1.IO.LoadFromFile('CRW_0001.CRW');
Cameraname := ImageEnView1.IO.Params.RAW_Camera;
```

DECLARATION

```
property RAW_ExtraParams: AnsiString;
```

DESCRIPTION

RAW_ExtraParams specifies extra parameters for external raw plugins (dcraw).

Example

```
ImageEnView1.IO.Params.RAW_ExtraParams := '-H 0 -
j';
ImageEnView1.IO.LoadFromFile('CRW_0001.CRW');
```

Unit ImageENIO
TIOParamVals

DECLARATION

```
property RAW_FourColorRGB: Boolean
```

DESCRIPTION

If RAW_FourColorRGB is true interpolates RGBG as four colors. The default is false.

Example

```
ImageEnView1.IO.Params.RAW_FourColorRGB:=true;
ImageEnView1.IO.LoadFromFile('CRW_0001.CRW');
```

DECLARATION

```
property RAW_Gamma: double;
```

DESCRIPTION

This property specifies the gamma used when loading the RAW. The default is 0.6.

Example

```
ImageEnView1.IO.Params.RAW_Gamma:=1;
ImageEnView1.IO.LoadFromFile('CRW_0001.CRW');
```

IOParamsVals

DECLARATION

```
property RAW_GetExifThumbnail: Boolean;
```

DESCRIPTION

Setting this property to true means that you want a thumbnail instead of the full image. This works only if a thumbnail is present in EXIF fields.

Example

```
// I want only the thumbnail
ImageEnView1.IO.Params.RAW_GetExifThumbnail :=
True;
ImageEnView1.IO.LoadFromFile('input.crw');
```

DECLARATION

```
property RAW_HalfSize: Boolean;
```

DESCRIPTION

If RAW_HalfSize is true specifies that we want load the RAW in half size. You can set this property to speed-up loading. The default is false.

Example

```
ImageEnView1.IO.Params.RAW_HalfSize := True;
ImageEnView1.IO.LoadFromFile('CRW_0001.CRW');
```

DECLARATION

```
property RAW_QuickInterpolate: Boolean;
```

DESCRIPTION

If RAW_QuickInterpolate is true uses quick, low-quality color interpolation. The default is true.

Example

```
ImageEnView1.IO.Params.RAW_QuickInterpolate :=
false;
ImageEnView1.IO.LoadFromFile('CRW_0001.CRW');
```

DECLARATION

```
property RAW_RedScale: double
```

DESCRIPTION

This property specifies the red multiplier used when loading the RAW. The default is 1.0 (daylight).

Example

ImageEnView1.IO.Params.RAW_RedScale:=0.8;
ImageEnView1.IO.LoadFromFile('CRW_0001.CRW');

IOParamsVals

DECLARATION

```
property RAW_UseAutoWB: Boolean;
```

DESCRIPTION

If RAW_UseAutoWB is true uses automatic white balance. The default is false.

Example

```
ImageEnView1.IO.Params.RAW_UseAutoWB:=true;
ImageEnView1.IO.LoadFromFile('CRW_0001.CRW');
```

DECLARATION

```
property RAW_UseCameraWB: Boolean
```

DESCRIPTION

If RAW_UseCameraWB is true uses camera white balance, if possible. The default is false.

Example

```
ImageEnView1.IO.Params.RAW_UseCameraWB:=true;
ImageEnView1.IO.LoadFromFile('CRW_0001.CRW');
```

MediaFile (AVI, MPEG, WMV..)

DECLARATION

```
property MEDIAFILE_FrameCount: integer;
```

DESCRIPTION

MEDIAFILE_FrameCount specifies the number of frames of last loaded media file (using OpenMediaFile).

DECLARATION

```
property MEDIAFILE_FrameDelayTime: integer;
```

DESCRIPTION

MEDIAFILE_FrameDelayTime is the time in milliseconds the image will be shown on playing.

RealRAW (not camera RAW)

DECLARATION

```
property BMPRAW_DataFormat: TIOBMPRAWDataFormat;
```

DESCRIPTION

BMPRAW_DataFormat specifies the next input/output data format. ASCII text values must be separated by one o more non-alpha characters (#13, #10, #32...).

Demo

Inputoutput/RealRAW

Example

```
// load a RAW image, 16 bit gray scale, ascii-
// text
ImageEnView1.LegacyBitmap := false;
ImageEnView1.IEBitmap.Allocate(1024, 768, ie16g);
ImageEnView1.IO.Params.BMPRAW_RowAlign := 8;
ImageEnView1.IO.Params.BMPRAW_HeaderSize := 0;
ImageEnView1.IO.Params.BMPRAW_DataFormat :=
dfTextDecimal;
ImageEnView1.IO.LoadFromFileBMPRAW('input.dat');
```

Unit ImageENIO
TIOParamVals

DECLARATION

```
property BMPRAW_ChannelOrder:
TIOBMPRAWChannelOrder;
```

DESCRIPTION

BMPRAW_ChannelOrder specifies the channels order for a RAW file. It can be coRGB or coBGR and it is valid only for color images.

Demo

Inputoutput/RealRAW

Example

```
// load a RAW image, RGB, interleaved, 8 bit
// aligned, 1024x768
ImageEnView1.LegacyBitmap:=false;
ImageEnView1.IEBitmap.Allocate(1024,768,ie24RGB);
ImageEnView1.IO.Params.BMPRAW_ChannelOrder:=coRGB;
ImageEnView1.IO.Params.BMPRAW_Planes:= plInterleaved;
ImageEnView1.IO.Params.BMPRAW_RowAlign:=8;
ImageEnView1.IO.Params.BMPRAW_HeaderSize:=0;
ImageEnView1.IO.LoadFromFileBMPRAW('input.dat');
// load a RAW image, CMYK, interleaved, 8 bit
// aligned, 1024x768
ImageEnView1.LegacyBitmap:=false;
ImageEnView1.IEBitmap.Allocate(1024,768,ieCMYK);
ImageEnView1.IO.Params.BMPRAW_ChannelOrder:=coRGB;
ImageEnView1.IO.Params.BMPRAW_Planes:= plInterleaved;
ImageEnView1.IO.Params.BMPRAW_RowAlign:=8;
ImageEnView1.IO.Params.BMPRAW_HeaderSize:=0;
ImageEnView1.IO.LoadFromFileBMPRAW('input.dat');
// saves current image as RAW image
ImageEnView1.IO.Params.BMPRAW_ChannelOrder:=coRGB;
ImageEnView1.IO.Params.BMPRAW_Planes:= plPlanar;
ImageEnView1.IO.Params.BMPRAW_RowAlign:=8;
ImageEnView1.IO.SaveToFileBMPRAW('output.dat');
```

IOParamsVals

DECLARATION

```
property BMPRAW_HeaderSize: integer;
```

DESCRIPTION

BMPRAW_HeaderSize specifies the header size in bytes. The header will be bypassed.

Demo
Inputoutput/RealRAW

DECLARATION

```
property BMPRAW_Planes: TIOBMPRAWPlanes;
```

DESCRIPTION

BMPRAW_Planes specifies how channels are disposed.

Demo
Inputoutput/RealRAW

DECLARATION

```
property BMPRAW_RowAlign: integer;
```

DESCRIPTION

BMPRAW_RowAlign specifies the row align in bits.

Demo
Inputoutput/RealRAW

Adobe XMP info

DECLARATION

```
property XMP_Info: AnsiString;
```

DESCRIPTION

Returns Adobe XMP info loaded from Jpeg, TIFF and PSD file formats. Adobe XMP is a XML coded string according to Adobe XMP specification updated to January 2004. ImageEn doesn't parse the XMP-XML string, so to read single tags you have to write code to parse the XML string.

XMP_Info is also saved back to Jpeg, TIFF and PSD file formats.

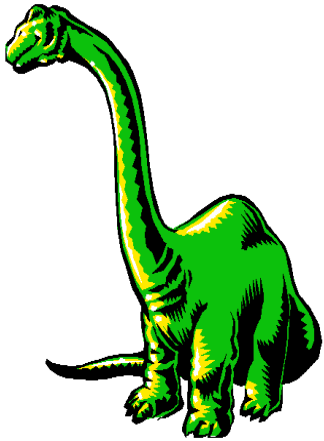
IOParamsVals

Chapter 8

TIEImagingAnnot

Chapter 8. IEImagingAnnot

DESCRIPTION



ImagingAnnot contains the (Wang) imaging annotations loaded (or to save) from a TIFF.

Using TIEImagingAnnot object you can create new objects, copy to a TImageEnVect (as vectorial objects), copy from a TImageEnVect (from vectorial objects) or just draw on the bitmap.

Unit ImageENIO

TIEImagingAnnot

Methods and Properties

Assign	Clear	CopyFromTImageEnVect
CopyToTImageEnVect	DrawToBitmap	LoadFromStandardBuffer
LoadFromStream	Objects	ObjectsCount
SaveToStandardBuffer	SaveToStream	

Example

```
// load an image and all annotations from
// input.tif . This allow to modify the
// annotations:
ImageEnVect1.IO.LoadFromFile('input.tif');
ImageEnVect1.IO.Params.ImagingAnnot.CopyToTImageE
nVect( ImageEnVect1 );

// load an image and all annotations from
// input.tif, but just draw on the image (for
// display):
ImageEnVect1.IO.LoadFromFile('input.tif');
ImageEnVect1.IO.Params.ImagingAnnot.DrawToBitmap(
ImageEnVect1.IEBitmap, true );
ImageEnVect1.Update;
```

Methods and Properties

DECLARATION

```
procedure Assign(Source: TIEImagingAnnot);
```

DESCRIPTION

Assign copies from another TIEImagingAnnot object.

DECLARATION

```
procedure Clear;
```

DESCRIPTION

Clear removes all annotations.

DECLARATION

```
procedure CopyFromTImageEnVect(Target:
TObject=nil);
```

DESCRIPTION

CopyFromTImageEnVect copies vectorial objects from a TImageEnVect object.
If Target is nil then the parent TImageEnVect is given.

Example

```
ImageEnVect1.IO.Params.ImagingAnnot.CopyFromTImag
eEnVect( ImageEnVect1 );
```

DECLARATION

```
procedure CopyToTImageEnVect(Target:
TObject=nil);
```

DESCRIPTION

CopyToTImageEnVect copies to a TImageEnVect object (as vectorial objects). If Target is nil then the parent TImageEnVect is given.

Example

```
ImageEnVect1.IO.Params.ImagingAnnot.CopyToTImageE
nVect( ImageEnVect1 );
```

DECLARATION

```
procedure DrawToBitmap(target: TIEBitmap;
Antialias: Boolean);
```

DESCRIPTION

DrawToBitmap draws all annotations to the specified bitmap.

Example

```
ImageEnView1.IO.ImagingAnnot.DrawToBitmap(
ImageEnView1.IEBitmap, true );
ImageEnView1.Update;
```

DECLARATION

```
procedure LoadFromStandardBuffer(buffer: pointer;
buflen: integer);
```

DESCRIPTION

LoadFromStandardBuffer loads imaging annotations from the specified buffer. Used when ImageEn loads image files, to extract imaging annotations.

DECLARATION

```
procedure LoadFromStream(Stream: TStream);
```

DESCRIPTION

For internal use only.

DECLARATION

```
property Objects[idx: integer]: TIEImagingObject;
```

DESCRIPTION

Objects contain the internal representation of an object.

DECLARATION

```
property ObjectsCount: integer;
```

DESCRIPTION

ObjectsCount is the objects (annotations) count.

DECLARATION

```
procedure SaveToStandardBuffer(var Buffer:
pointer; var BufferLength: integer);
```

DESCRIPTION

SaveToStandardBuffer saves imaging objects to buffer. You have to free the buffer. This method is used to embed imaging annotations inside image files.

DECLARATION

```
procedure SaveToStream(Stream: TStream);
```

DESCRIPTION

For internal use only.

Chapter 9

TImageEnMIO

Chapter 9. ImageEnMIO

TImageEnMIO loads/saves and acquires multi-images. It must be attached to TImageEnMView. TImageEnMIO is used for all graphic input and output.

TImageEnMIO handles input/output operations.

TImageEnMView (ImageEnMView1) already encapsulates TImageEnMIO and TImageEnProc, so you don't actually need to put a TImageEnMIO or a TImageEnProc component on the form.

TImageEnMIO must be attached to a TImageEnView (or inherited objects), TIEBitmap, TImage and TBitmap objects. TImageEnMIO can be accessed with ImageEnMView1.MIO instead of using the ImageEnMIO1 component and ImageEnProc can be accessed with ImageEnMView1.Proc instead of using the ImageEnProc1 component.

TImageEnMIO can load animated GIF, multi-image TIFF, AVI and acquire multi-page from TWain scanners. It can also save animated GIF, multi-images TIFF and AVI.

Unit ImageEnMIO

TImageEnMIO

Note: Users often attach a TImageEnIO component to TImageEnMView component. This is **not** correct. The TImageEnMIO component must be attached only to a TImageEnMView component.

Methods and Properties

TWain/WIA scanner

AcquireClose	AcquireOpen
Acquire	SelectAcquireSource
TWainParams	WIAParams

Unit ImageEnMIO
TImageEnMIO

Generic input/output

Aborting	AllowMalformedPages	AutoAdjustDPI
AttachedMView	DefaultDitherMethod	DuplicateCompressionInfo
ExecuteOpenDialog	ExecuteSaveDialog	FilteredAdjustDPI
LoadFromBuffer	LoadFromFileAuto	LoadFromFileFormat
LoadFromFile	LoadFromFiles	LoadFromStreamFormat
LoadFromStream	LoadFromURL	NativePixelFormat
Params	ParamsCount	ProxyAddress
ProxyPassword	ProxyUser	SaveToFile
Update		

Dialogs

DialogsMeasureUnit	DoPreviews	MsgLanguage
PreviewFont	PreviewsParams	SimplifiedParamsDialogs

Printing

DoPrintPreviewDialog	PrintImage	PrintImagePos
PrintImages	PrintingFilterOnSubsampling	
PrintPreviewParams		

Unit ImageEnMIO
TImageEnMIO

AVI

LoadFromFileAVI

SaveToFileAVI

DCX

LoadFromFileDCX

LoadFromStreamDCX

SaveToFileDCX

SaveToStreamDCX

GIF

LoadFromFileGIF

LoadFromStreamGIF

SaveToFileGIF

SaveToStreamGIF

ICO

LoadFromFileICO

LoadFromStreamICO

SaveToFileICO

SaveToStreamICO

TIFF

LoadFromFileTIFF

LoadFromStreamTIFF

SaveToFileTIFF

SaveToStreamTIFF

DirectShow Media Files

LoadFromMediaFile

Unit ImageEnMIO
TImageEnMIO

Adobe PDF

SaveToFilePDF
SaveToStreamPDF

PostScript

SaveToFilePS
SaveToStreamPS

DICOM Medical imaging

LoadFromFileDICOM
LoadFromStreamDICOM

Events

OnAcquireBitmap
OnAfterAcquireBitmap
OnDoPreviews
OnFinishWork
OnProgress

Methods and Properties

Twain-WIA Scanner

DECLARATION

```
procedure AcquireClose;
```

DESCRIPTION

AcquireClose closes a connection opened with AcquireOpen. It is useful to do a modeless acquisition. Whenever ImageEn gets an image, the OnAcquireBitmap and OnAfterAcquireBitmap event occur.

DECLARATION

```
function AcquireOpen: Boolean;
```

DESCRIPTION

AcquireOpen opens a connection to the selected scanner. It is useful to do a modeless acquisition. AcquireOpen returns false if fail. Whenever ImageEn gets an image the OnAcquireBitmap and OnAfterAcquireBitmap events occur.

DECLARATION

```
function Acquire(api: TIEAcquireApi): Boolean;
```

DESCRIPTION

Perform multi-page image acquisition from TWain or WIA devices. Returns True if the acquisition has succeeded, False otherwise.

DECLARATION

```
function SelectAcquireSource(api: TIEAcquireApi):
Boolean;
```

DESCRIPTION

SelectAcquireSource shows a dialog where the user can select a TWain or WIA device.

SelectAcquireSource returns false if user press “Cancel” button.

Unit ImageEnMIO
TImageEnMIO

DECLARATION

```
property TWainParams: TIETWainParams;
```

DESCRIPTION

Use the TWainParams property to control scanner acquisition. You can enable/disable standard user interface, set pixeltype (Grayscale, RGB...), DPI, or select the acquire scanner without user interaction. Look at TIETWainParams object for more details.

Example

```
// Acquires a black/white (1bit) image
ImageEnMView1.MIO.TWainParams.PixelType.CurrentValue:
=0;
ImageEnMView1.MIO.TWainParams.VisibleDialog:=False;
ImageEnMView1.MIO.Acquire;
```

DECLARATION

```
property WIAParams: TIEWia;
```

DESCRIPTION

WIAParams allows you to set/get parameters, show dialogs and control WIA devices. Look at TIEWia for more info.

Generic input/output

DECLARATION

```
property Aborting: Boolean;
```

DESCRIPTION

Applications can abort save/load processing by assigning True to the Aborting property. On loading, the image will be truncated. On saving, the file will be closed and truncated (and will be unreadable). You can also read the Aborting property to know when aborting is in progress.

Unit ImageEnMIO

TImageEnMIO

DECLARATION

```
property AllowMalformedPages: Boolean;
```

DESCRIPTION

If true malformed pages (for example loading a corrupted or out of standard TIFF), if possible, will be loaded anyway.
Default is False (stop loading).

DECLARATION

```
property AutoAdjustDPI: Boolean;
```

DESCRIPTION

When AutoAdjustDPI is True and last loaded/scanned image has horizontal DPI not equal to vertical DPI, ImageEn resizes the image making DPIX=DPIY. The default is False.

DECLARATION

```
property AttachedMView: TImageEnMView;
```

DESCRIPTION

This property specifies the attached TImageEnMView component. You must set this property to use TImageEnMIO. ImageEnMView.IO may also be used if not using the TImageEnMView component.

DECLARATION

```
property DefaultDitherMethod: TIEDitherMethod;
```

DESCRIPTION

DefaultDitherMethod specifies the default dithering method to apply when a color image needs to be converted to black/white. IeThreshold is the default.

DECLARATION

```
procedure DuplicateCompressionInfo;
```

DESCRIPTION

DuplicateCompressionInfo clones the compression information of page 0 to all pages.

Unit ImageEnMIO
TImageEnMIO

DECLARATION

```
function ExecuteOpenDialog(InitialDir: string;
InitialFileName: string; AlwaysAnimate: Boolean;
FilterIndex: integer; ExtendedFilters: string;
MultiSelect: Boolean; Title: WideString; Filter:
WideString=''): string;
```

DESCRIPTION

The ExecuteOpenDialog shows and executes the open dialog. It encapsulates the TOpenImageEnDialog component. InitialDir is the starting directory. InitialFileName is the default file name with extension. If AlwaysAnimate is true then auto-animate GIF and AVI.

FilterIndex specifies what file format to select for default. Allowed values:

1 : Common graphics formats
2 : All files
3 : TIFF Bitmap (TIF;TIFF;FAX)
4 : GIF (GIF) - if enabled5 : JPEG Bitmap (JPG;JPEG;JPE)
6 : PaintBrush (PCX)
7 : Windows Bitmap (BMP;DIB;RLE)
8 : Windows Icon (ICO)
9 : Windows Cursor (CUR)
10 : Portable Network Graphics (PNG)
11 : Windows Metafile (WMF)
12 : Enhanced Windows Metafile (EMF)
13 : Targa Bitmap (TGA;TARGA;VDA;ICB;VST;PIX)
14 : Portable Pixmap, GreyMap, BitMap (PXM;PPM;PGM;PBM)
15 : Wireless bitmap (WBMP)
16 : Jpeg2000 (JP2)
17 : Jpeg2000 Code Stream (J2K;JPC;J2C)
18 : Multipage PCX (DCX)
19 : Camera RAW (RAW;CR2;CRW...)
20 : Video for Windows (AVI)
21 : Video Mpeg (MPEG)
22 : Windows Media Video (WMV)

ExtendedFilters specifies additional file formats (example: 'Fun Bitmap|*.fun;*.fan'). If MultiSelect is True then it is possible to select multiple files. The returned string contains a list of filename separated by '|' character ('one.jpg|two.jpg'). Title specifies the dialog title. Empty string means operating system default value. Filter specifies file extensions to enable (i.e. 'JPEG Bitmap (JPG)|*.jpg|CompuServe Bitmap (GIF)|*.gif').

ImageEnMIO

ExecuteOpenDialog returns null string (‘’) if the user clicks on Cancel.

DECLARATION

```
function ExecuteSaveDialog(InitialDir: string;
InitialFileName: string; AlwaysAnimate: Boolean;
FilterIndex: integer; ExtendedFilters: string;
Title: WideString; Filter: WideString): string;
```

DESCRIPTION

The ExecuteSaveDialog shows and executes the save dialog. It encapsulates the TSaveImageEnDialog component.

InitialDir is the starting directory.

InitialFileName is the default file name with extension.

AlwaysAnimate if true auto-animate GIF and AVI
FilterIndex specifies what file format to select by default.

The InitialFileName extension, if specified, in the second parameter, will over-ride the specified index. Valid values are:

1 : TIFF Bitmap (TIF;TIFF;FAX)
2 : GIF (GIF) - if enabled
3 : JPEG Bitmap (JPG;JPEG;JPE)
4 : PaintBrush (PCX)
5 : Windows Bitmap (BMP;DIB;RLE)
6 : Windows Icon (ICO)
7 : Portable Network Graphics (PNG)
8 : Targa Bitmap (TGA;TARGA;VDA;ICB;VST;PIX)
9 : Portable Pixmap, GreyMap, BitMap (PXM;PPM;PGM;PBM)
10 : Wireless Bitmap (WBMP)
11 : Jpeg2000 (JP2)
12 : Jpeg2000 Code Stream (J2K;JPC;J2C)
13 : PostScript Level 2 (PS;EPS)
14 : Adobe PDF (PDF)
15 : Multipage PCX (DCX)
16 : Video for Windows (AVI)

ExtendedFilters specifies additional file formats (example: 'Fun Bitmap|*.fun;*.fan')

Title specifies the dialog title. Empty string means operating system default value.

Filter specifies file extensions to enable (i.e. 'JPEG Bitmap (JPG)|*.jpg|CompuServe Bitmap (GIF)|*.gif').

ExecuteSaveDialog returns a null string (") if the user clicks on Cancel.

Unit ImageEnMIO

TImageEnMIO

DECLARATION

```
property FilteredAdjustDPI: Boolean;
```

DESCRIPTION

The FilteredAdjustDPI property is valid when AutoAdjustDPI is true. If set to True, ImageEn applies a resampling filter to the image to enhance quality. It can slow down the loading process.

DECLARATION

```
procedure LoadFromBuffer(Buffer: pointer;
BufferSize: integer; Format: TIOFileType =
ioUnknown);
```

DESCRIPTION

LoadFromBuffer loads an image or a multipage image from the specified buffer.

Parameter	Description
Buffer	The buffer pointer.
BufferSize	The buffer length in bytes.
Format	Specifies the expected file format. If Format is ioUnknown, then try to find the format automatically.

ImageEnMIO

Declaration

```
procedure LoadFromFileAuto(const FileName:
string);
```

Description

LoadFromFileAuto loads an image from file. To detect file format it doesn't look at the file extension, but at the file content.

Declaration

```
procedure LoadFromFileFormat(const FileName:
string; FileFormat: TIOFileType);
```

Description

LoadFromFileFormat loads an image from file. LoadFromFileFormat recognizes the file format from FileFormat parameter.

If the file does not represent a valid image format, the Aborting property is true. FileName is the file name.

Unit ImageEnMIO
TImageEnMIO

DECLARATION

```
procedure LoadFromFile(const FileName: string);
```

DESCRIPTION

LoadFromFile loads a multi-image file. It detects the file format from the file name extension. The source can be also an URL, if it has the form 'http://'. If the file does not represent a valid image format, the Aborting Property is true. FileName is the file name with extension.

DECLARATION

```
procedure LoadFromFiles(const FileName: string;
AutoDetect: Boolean=false; LoadWhenViewed:
Boolean=false);
```

DESCRIPTION

Loads multiple files separated by '|' character. If AutoDetect is true then ImageEn tries to detect file type from header, otherwise it looks only the file extension. If LoadWhenViewed is true each file is actually loaded only when it needs to be displayed.

DECLARATION

```
procedure LoadFromStreamFormat(Stream: TStream;
FileFormat: TIOFileType);
```

DESCRIPTION

LoadFromStreamFormat loads an image (or multiple images) from the specified stream assuming that the image has FileFormat format.

DECLARATION

```
procedure LoadFromStream(Stream: TStream);
```

DESCRIPTION

LoadFromStream loads an image (or multiple images) from the specified stream. LoadFromStream tries to detect file format reading the image header.

Unit ImageEnMIO
TImageEnMIO

DECLARATION

```
procedure LoadFromURL(URL: string);
```

DESCRIPTION

LoadFromURL loads a multipage file from the network using the HTTP or FTP protocol, specifying the URL. This function doesn't support password authentication for HTTP, while it is necessary for FTP (also connecting to anonymous server).

URL must have the syntax:
'http://domain[:port]/resource'
'https://domain[:port]/resource'
'ftp://user:password@domain[:port]/resource'

It is possible to set proxy parameters using ProxyAddress, ProxyUser and ProxyPassword properties.

DECLARATION

```
property NativePixelFormat: Boolean;
```

DESCRIPTION

By setting this property to True, you disable the conversion of paletted images and gray scale to 24 bit. By default, ImageEn converts all paletted images to 24 bit (true color). Only black/white images are stored in the original format with 1 bit per pixel. Note that setting NativePixelFormat=True, you will not be able to execute some image processing operations.

DECLARATION

```
property Params[idx: integer]: TIOParamsVals;
```

DESCRIPTION

This property contains a TIOParamsVals object for the idx image. This one contains some parameters (as bits per sample, type of compression, etc.) of each image file format. The parameters are updated when you load files or streams. Also you can modify these parameters before saving images.

Read-only

Unit ImageEnMIO

TImageEnMIO

DECLARATION

```
property ParamsCount: integer;
```

DESCRIPTION

ParamsCount is the number of elements in the Params property (alias number of images contained in TImageEnMView attached object).

Read-only

DECLARATION

```
property ProxyAddress: string;
```

DESCRIPTION

ProxyAddress specifies the proxy address and port when using LoadFromURL method. The syntax is: ‘domain:port’.

DECLARATION

```
property ProxyPassword: string;
```

DESCRIPTION

ProxyPassword specifies the proxy password when using LoadFromURL method.

DECLARATION

```
property ProxyUser: string;
```

DESCRIPTION

ProxyUser specifies the proxy userid when using LoadFromURL method.

DECLARATION

```
procedure SaveToFile(const FileName: string);
```

DESCRIPTION

SaveToFile save a multi-image file as a GIF, TIFF, DCX, ICO, and AVI. It detects the file's format from the extension. FileName is the file name with extension.

DECLARATION

```
procedure Update;
```

DESCRIPTION

Update calls the Update method of the attached TImageEnMView component.

Dialogs

Declaration

```
property DialogsMeasureUnit:
TIEDialogsMeasureUnit
```

Description

The DialogsMeasureUnit property specifies the measurement unit used in the print preview dialog (see DoPrintPreviewDialog).

Declaration

```
function DoPreviews(idx: integer; pp:
TPreviewParams): Boolean;
```

Description

DoPreviews executes the Previews dialog for the image idx. This dialog gets/sets the parameters of image file formats. pp is the set of the image format parameters to be displayed by the dialog. If idx is -1, the dialogs act on all images.

Unit ImageEnMIO

TImageEnMIO

DECLARATION

```
property MsgLanguage: TMsgLanguage;
```

DESCRIPTION

This property sets the language for Input/output previews (IOPreviews) dialogs (Previews).

DECLARATION

```
property PreviewFont: TFont;
```

DESCRIPTION

The PreviewFont property contains the font used in the Previews dialog. Make sure the size of the font match label's length.

DECLARATION

```
property PreviewsParams: TIOPreviewsParams;
```

DESCRIPTION

This property specifies the features that the input/output preview dialog will have.

DECLARATION

```
property SimplifiedParamsDialogs: Boolean;
```

DESCRIPTION

If the SimplifiedParamsDialogs property is true (the default), the 'Advanced' button of open/save dialogs will show a simplified set of parameters. Warning: the default is true, to allow old style "advanced" dialogs set it to False.

Printing

DECLARATION

```
function DoPrintPreviewDialog(const TaskName: string; PrintAnnotations:  Boolean; const Caption:  string; ThumbnailPrinting: Boolean): Boolean;
```

DESCRIPTION

DoPrintPreviewDialog executes the multi-page print preview dialog. This function is like DoPrintPreviewDialog of TImageEnIO, but allows working with multiple pages. If ThumbnailPrinting is true then it defaults to printing of thumbnails, otherwise the default is read from PrintPreviewParams. If PrintAnnotation is true and the image contains Imaging Annotations they will be printed out. Caption specifies the caption of preview dialog.

See also PrintPreviewParams.

ImageEnMIO

Unit ImageEnMIO
TImageEnMIO

DECLARATION

```
procedure PrintImage(ImageIndex: integer;
PrtCanvas: TCanvas; MarginLeft, MarginTop,
MarginRight, MarginBottom:  double; VerticalPos:
TIEVerticalPos;  HorizontalPos:
TIEHorizontalPos;  Size: TIESize; SpecWidth,
SpecHeight: double; GammaCorrection: double);
```

DESCRIPTION

PrintImage prints the specified image by specifying margins, vertical position, horizontal position and size.

ImageIndex is the index of the page/image to print (0=first image). PrtCanvas is the printing canvas. The application can set this parameter from Printer.Canvas. MarginLeft, MarginTop, MarginRight, MarginBottom are the page margins in inches. By specifying all zero values, no margins are used. VerticalPos determines how the image vertically aligns within the page. HorizontalPos determines how the image horizontally aligns within the page. Size determines the size of the image. SpecWidth, SpecHeight specify the absolute sizes of the image in inches. Valid only when Size=iesSPECIFIEDSIZE.

GammaCorrection is the gamma correction value, use 1 to disable gamma correction.

Unit ImageEnMIO
TImageEnMIO

DECLARATION

```
procedure PrintImagePos(ImageIndex: integer;
PrtCanvas: TCanvas; x,y: double; Width, Height:
double; GammaCorrection: double);
```

DESCRIPTION

PrintImagePos prints the specified image, specifying absolute position and size.

PrtCanvas is the printing canvas. The application can set this parameter from Printer.Canvas.

ImageIndex is the page/image index (0=first page/image).
x,y is the top-left starting point, in inches. Width, Height is the image size, in inches. GammaCorrection is the gamma correction value. Use a value of 1.0 to disable gamma correction.

Unit ImageEnMIO

TImageEnMIO

DECLARATION

```
procedure PrintImages(Columns: integer; Rows:
integer; HorizSpace: double; VertSpace: double;
PrintSelected: Boolean; MarginLeft: double;
MarginTop: double; MarginRight: double;
MarginBottom: double; DrawBox: Boolean; DrawText:
Boolean; DrawShadow: Boolean; BoxColor: TColor =
clBlack);
```

DESCRIPTION

PrintImages prints all images or only selected images.
You can specify number of columns and rows, spaces between images and margins. It is possible also to draw box around images, shadows and text.

Parameter	Description
Columns	Specifies how arrange images, specifying the number of columns.
Rows	Specifies how arrange images, specifying the number of rows.
HorizSpace	The horizontal space in inches between images.
VertSpace	The vertical space in inches between images.
PrintSelected	This is true if you want to print only selected images.
MarginLeft	Page left margin in inches. By specifying all zero values, no margins are used.
MarginTop	Page top margin in inches. By specifying all zero values, no margins are used.

Parameter	Description
MarginRight	Page right margin in inches. By specifying all zero values, no margins are used.
MarginBottom	Page bottom margin in inches. By specifying all zero values, no margins are used.
DrawBox	This is true if you want to draw a box around the images (image space). Image is always stretched to maintain aspect ratio.
DrawText	This is true if you want to draw text associated with every image.
DrawShadow	This is true if you want a shadow around the image.
BoxColor	Specifies the color of the box around the image if DrawBox is True.

DECLARATION

```
property PrintingFilterOnSubsampling:
TResampleFilter;
```

DESCRIPTION

PrintingFilterOnSubsampling specifies a filter when the image needs to be printed and it must be resampled. Filtering enhances the image quality but slows processing.

DECLARATION

```
property PrintPreviewParams:
TIOPrintPreviewParams;
```

DESCRIPTION

This property allows you to set/get parameters of print preview dialog. All measure units are specified by DialogsMeasureUnit property.

AVI

DECLARATION

```
procedure LoadFromFileAVI(const FileName:
string);
```

DESCRIPTION

Use LoadFromFileAVI to cause loading of an AVI file.

If the file does not represent a valid image format, the Aborting Property is true. FileName is the file name with extension.

Unit ImageEnMIO
TImageEnMIO

DECLARATION

```
procedure SaveToFileAVI(const FileName:string;
const codec: AnsiString);
```

DESCRIPTION

SaveToFileAVI saves a multi-image file as AVI format. Images in TImageEnMView must be of the same size. FileName is the file name with extension.

Codec specifies the compression codec to use (must be installed on system) as four characters length string.
Example:
'cvid' : cinepak by Radius
'msvc' : Microsoft Video 1
'mp42' : Microsoft MPEG4 V2

More codecs at www.fourcc.org or Microsoft (or at msdn.microsoft.com searching for registered fourcc codes and wave formats).

If codec is not specified (or if it is an empty string), a dialog box appears to allow the user to select a compression.

DCX

DECLARATION

```
procedure LoadFromFileDCX(const FileName:string);
```

DESCRIPTION

Loads all images contained in a multi-image DCX file.

DECLARATION

```
procedure LoadFromStreamDCX(Stream: TStream);
```

DESCRIPTION

Loads all images contained in a multi-image DCX stream.

DECLARATION

```
procedure SaveToFileDCX(const FileName:string);
```

DESCRIPTION

SaveToFileDCX saves all images in DCX format to the specified file.

DECLARATION

```
procedure SaveToStreamDCX(Stream: TStream);
```

DESCRIPTION

SaveToStreamDCX saves all images in DCX format to the specified stream.

GIF

DECLARATION

```
procedure LoadFromFileGIF(const FileName:
string);
```

DESCRIPTION

Use LoadFromFileGIF to cause loading a GIF file. If the file does not represent a valid image format, the Aborting property is true. FileName is the file name with extension.

DECLARATION

```
procedure LoadFromStreamGIF(Stream:  TStream);
```

DESCRIPTION

LoadFromStreamGIF loads a GIF from a stream. LoadFromStreamGIF inserts the image from SelectedImage. If the stream does not represent a valid image format, the Aborting property is true.

DECLARATION

```
procedure SaveToFileGIF(const FileName:  string);
```

DESCRIPTION

SaveToFileGif force to save a multi-image file as GIF. FileName is the file name with extension.

DECLARATION

```
procedure SaveToStreamGIF(Stream:  TStream);
```

DESCRIPTION

SaveToStreamGif saves a GIF to a stream.

ICO

DECLARATION

```
procedure LoadFromFileICO(const FileName:
string);
```

DESCRIPTION

Load all images contained in a multi-image ICO file.

DECLARATION

```
procedure LoadFromStreamICO(Stream: TStream);
```

DESCRIPTION

Load all images contained in a multi-image ICO stream.

DECLARATION

```
procedure SaveToFileICO(const FileName:string);
```

DESCRIPTION

SaveToFileICO saves all images in ICO format to the specified file.

ImageEnMIO

DECLARATION

```
procedure SaveToStreamICO(Stream: TStream);
```

DESCRIPTION

SaveToStreamICO saves all images in ICO format to the specified stream.

TIFF

DECLARATION

```
procedure LoadFromFileTIFF(const FileName: string);
```

DESCRIPTION

Use LoadFromFileTIFF to cause loading a TIFF file. If the file does not represent a valid image format, the Aborting property is true. FileName is the file name with extension.

DECLARATION

```
procedure LoadFromStreamTIFF(Stream: TStream);
```

DESCRIPTION

Load a TIFF from a stream. If the stream does not represent a valid image format, the Aborting property is true.

DECLARATION

```
procedure SaveToFileTIFF(const FileName:string;
SaveSelected: Boolean=false);
```

DESCRIPTION

SaveToFileTIFF saves a multi-image file as TIFF format. FileName is the file name with extension. Save only selected pages if SaveSelected is true.

DECLARATION

```
procedure SaveToStreamTIFF(Stream: TStream;
SaveSelected: Boolean=false);
```

DESCRIPTION

SaveToStreamTIFF saves a TIFF to a stream. Save only selected pages if SaveSelected is true.

ImageEnMIO

Unit ImageEnMIO

TImageEnMIO

DirectShow Media Files

DECLARATION

```
procedure LoadFromMediaFile(const FileName:
string); dynamic;
```

DESCRIPTION

LoadFromMediaFile loads a video using DirectShow. When LoadFromFileAVI fails, you should try this method. Also use LoadFromMediaFile to load DirectShow supported files, like wmv, mpeg, avi.

ImageEnMIO

Unit ImageEnMIO

TImageEnMIO

Adobe PDF

DECLARATION

```
procedure SaveToFilePDF(const FileName:string);
```

DESCRIPTION

SaveToFilePDF creates a multipage Adobe PDF file with all images loaded within the attached TImageEnMView component.

DECLARATION

```
procedure SaveToStreamPDF(Stream: TStream);
```

DESCRIPTION

SaveToStreamPDF creates a multipage Adobe PDF file with all images loaded within the attached TImageEnMView component.

ImageEnMIO

PostScript

DECLARATION

```
procedure SaveToFilePS(const FileName: string);
```

DESCRIPTION

SaveToFilePS creates a multi-page PostScript file with all images loaded in the attached TImageEnMView component.

DECLARATION

```
procedure SaveToStreamPS(Stream: TStream);
```

DESCRIPTION

SaveToStreamPS creates a multi-page PostScript file with all images loaded in the attached TImageEnMView component.

ImageEnMIO

Unit ImageEnMIO

TImageEnMIO

DICOM Medical Imaging

DECLARATION

```
procedure LoadFromFileDICOM(const FileName: string);
```

DESCRIPTION

LoadFromFileDICOM loads a DICOM image or a multipage DICOM. This method is necessary when the DICOM file hasn't extension and hasn't a valid DICOM header, but you are sure that is a DICOM file. DICOM parameters are stored in DICOM_Tags.

DECLARATION

```
procedure LoadFromStreamDICOM(Stream: TStream);
```

DESCRIPTION

LoadFromStreamDICOM loads a DICOM image or a sequence of images from stream. DICOM parameters are stored in DICOM_Tags.

ImageEnMIO

Events

DECLARATION

```
property OnAcquireBitmap: TIEAcquireBitmapEvent;
```

DESCRIPTION

OnAcquireBitmap occurs whenever the scanner acquires an image.

DECLARATION

```
property OnAfterAcquireBitmap:
TIEAfterAcquireBitmapEvent;
```

DESCRIPTION

OnAfterAcquireBitmap occurs just after an image acquired from scanner and hence added to the image list.

DECLARATION

```
property OnAcquireClose: TNotifyEvent;
```

DESCRIPTION

This event occurs when the user closes the acquire dialog, open using TwainAcquireOpen.

ImageEnMIO

DECLARATION

```
property OnDoPreviews: TIEDoPreviewsEvent;
```

DESCRIPTION

OnDoPreviews occurs just before DoPreviews is called or "Advanced" button is clicked (save dialog). You can avoid displaying the dialog setting Handled parameter True so to display a custom dialog.

DECLARATION

```
property OnFinishWork: TNotifyEvent;
```

DESCRIPTION

OnFinishWork occurs when an input/output task ends. It is useful in resetting progress bars, or to know when a thread ends in a asynchronous mode.

DECLARATION

```
property OnProgress: TIEProgressEvent;
```

DESCRIPTION

OnProgress event is called upon input/output operations.

Chapter 10

TIEWicReader

Chapter 10. TIEWICReader

The TIEWICReader class encapsulates some Microsoft Windows Imaging Component (WIC) interfaces and allows loading Microsoft HD Photo and other WIC installed file formats.

WIC preinstalled decoders are TIFF, PNG, GIF, ICO, BMP, JPEG, HDP. TIEWICReader requires Windows XP (SP2) with .Net 3.0, Windows Vista.

Methods and Properties

Close	DPIX	DPIY
FrameCount	FrameHeight	FrameWidth
GetFrame	IsAvailable	Open

Methods and Properties

DECLARATION

```
procedure Close;
```

DESCRIPTION

Closes currently open stream or file freeing allocated resources. This method is implicitly called by the free method.

DECLARATION

```
property DPIX: double;
```

DESCRIPTION

X-axis dots per inch (dpi) resolution. This property is filled by GetFrame method.

See also
DPIY

WicReader

Declaration

```
property DPIY: double;
```

Description

Y-axis dots per inch (dpi) resolution. This property is filled by GetFrame method.

See also
DPIX

Declaration

```
property FrameCount: integer;
```

Description

FrameCount returns the total number of frames in the image.

Declaration

```
function FrameHeight: integer;
```

Description

FrameHeight retrieves the height of the frame.

Unit IEWic

TIEWICReader

Declaration

```
function FrameWidth: integer;
```

Description

FrameWidth retrieves the width of the frame.

Declaration

```
procedure GetFrame(frameIndex: integer;
destBitmap: TIEBitmap; IOParams: TIOParamsVals =
nil; Aborting: Boolean = nil);
```

Description

GetFrame retrieves the specified frame of the image.
If IOParams is specified then it is filled also with EXIF metatags.

Look also

Open.

WicReader

Unit IEWic
TIEWICReader

Declaration

```
function IsAvailable: Boolean;
```

Description

Returns true if WIC is available. This should happen on Windows XP (SP2) with .Net 3.0, Windows Vista. Look also IEWICAvailable function.

Declaration

```
function Open(Stream: TStream; fileType:
TIOFileType): Boolean;
function Open(const FileName: WideString;
fileType: TIOFileType): Boolean;
```

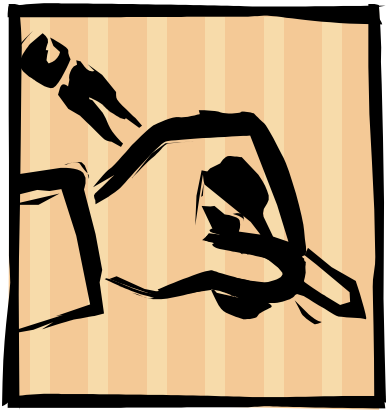
Description

Open opens a stream or file, enabling subsequent calls to GetFrame. You should call Free or Close in order to terminate reading.

Chapter 11

TIEWicWriter

Chapter 11. TIEWicWriter



The TIEWICWriter class encapsulates some Microsoft Windows Imaging Component (WIC) interfaces and allows writing Microsoft HD Photo and other WIC installed file formats.

WIC preinstalled encoders are TIFF, PNG, GIF, BMP, JPEG, HDP. TIEWICWriter requires Windows XP (SP2) with .Net 3.0, Windows Vista.

Unit IEWic

TIEWICWriter

Methods and Properties

Close	DPIX	DPIY
IsAvailable	Open	PutFrame

***Canonical* Encoder Parameter Properties**

CompressionQuality	ImageQuality	Lossless

Specific HD Photo Encoder Parameter Properties

FrequencyOrder	HorizontalTileSlices	Overlap
Quality	Subsampling	UseCodecOptions
VerticalTileSlices		

Methods and Properties

DECLARATION

```
procedure Close;
```

DESCRIPTION

Commit changes to output stream or file.
This method is implicitly called by the Free method.

DECLARATION

```
property DPIX: double;
```

DESCRIPTION

X-axis dots per inch (dpi) resolution. You have to set this property before each PutFrame call.

See also
DPIY

WicWriter

DECLARATION

```
property DPIY: double;
```

DESCRIPTION

Y-axis dots per inch (dpi) resolution. You have to set this property before each PutFrame call.

See also DPIX

DECLARATION

```
function IsAvailable: Boolean;
```

DESCRIPTION

Returns true if WIC is available. This should happen on Windows XP (SP2) with .Net 3.0, Windows Vista. Look also IEWICAvailable function.

WicWriter

Unit IEWic
TIEWICWriter

Declaration

```
function Open(Stream: TStream; fileType:
TIOFileType): Boolean;
function Open(const FileName: WideString;
fileType: TIOFileType):Boolean;
```

Description

Creates specified image format in stream or file, enabling subsequent calls to PutFrame calls. You should call Free or Close in order to commit changes.

WicWriter

Unit IEWic
TIEWICWriter

DECLARATION

```
procedure PutFrame(srcBitmap: TIEBitmap;
IOParams: TIOParamsVals = nil);
```

DESCRIPTION

PutFrame adds a new frame to current open stream or file. You can call multiple PutFrame only if the writing file format accepts more than one frame. If IOParams is specified parameters (DPI, compression) are read from it. PutFrame doesn't write EXIF metatags: you have to use InjectTIFFEXIF to inject EXIF in HDPhoto or TIFF files. It is important that you close the stream/file before inject EXIF metatags.

WicWriter

Canonical Encoder Parameter Properties

DECLARATION

```
property CompressionQuality: double;
```

DESCRIPTION

0.0 specifies the least efficient compression scheme available, typically resulting in a fast encode but larger output. A value of 1.0 specifies the most efficient scheme available, typically taking more time to encode but producing smaller output. HD Photo does not support this encoder option. You have to set this property before each PutFrame call.

DECLARATION

```
property ImageQuality: double;
```

DESCRIPTION

0.0 specifies the lowest possible fidelity rendition and 1.0 specifies the highest fidelity, which for HD Photo results in mathematically lossless compression. You have to set this property before each PutFrame call.

WicWriter

Unit IEWic
TIEWICWriter

DECLARATION

```
property Lossless: Boolean;
```

DESCRIPTION

Setting this parameter to true enables mathematically lossless compression mode and overrides the ImageQuality parameter setting. The default value is false. You have to set this property before each PutFrame call.

Specific HD Photo encoder parameter properties

DECLARATION

```
property FrequencyOrder:  Boolean;
```

DESCRIPTION

This parameter specifies that the image must be encoded in frequency order, with the lowest frequency data appearing first in the file, and image content grouped by its frequency rather than its spatial orientation. Organizing a file by frequency order provides the highest performance results for any frequency-based decoding, and is the preferred option. Device implementations of HD Photo encoders may choose to organize a file in spatial order to reduce the memory footprint required during encoding. The default value is true and it is recommended that applications and devices always use frequency order unless there is performance or application-specific reasons to use spatial order.

You have to set this property before each PutFrame call.

Unit IEWic
TIEWICWriter

DECLARATION

```
property HorizontalTileSlices: integer;
```

DESCRIPTION

HorizontalTileSlices and VerticalTileSlices specify the horizontal and vertical tiling of the image prior to compression encoding for the most optimal region decode performance. Dividing the image into rectangular tiles during encoding makes it possible to decode regions of the image without the need to process the entire compressed data stream. The default value of 0 specifies no subdivision, so the entire image is treated as a single tile. A value of 1 for each parameter will create a single horizontal and a single vertical division, effectively dividing the image into four equally sized tiles. The maximum value of 4095 for each parameter divides the image into 4096 tile rows with 4096 tiles per row. In other words, the parameter values equal the number of horizontal and vertical tiles (respectively) minus 1.
A tile can never be smaller than 16 pixels in width or height, so the HD Photo encoder may adjust this parameter to maintain the required minimum tile size. Because there is storage and processing overhead associated with each tile, these values should be chosen carefully to meet the specific scenario and unless there is a very specific reason, large numbers of small tiles should be avoided.

The default value for both parameters is 0.

You have to set this property before each PutFrame call.

WicWriter

DECLARATION

```
property Overlap: integer;
```

DESCRIPTION

This parameter selects the optional overlap processing level:

Value	Description
0	No overlap processing is enabled.
1	One level of overlap processing is enabled, modifying 4x4 block encoded values based on values of neighboring blocks.
2	Two levels of overlap processing are enabled; in addition to the first level processing, encoded values of 16x16 macro blocks are modified based on the values of neighboring macro blocks.

The default value is 1.

UseCodecOptions must be true.
You have to set this property before each PutFrame call.

WicWriter

Unit IEWic
TIEWICWriter

DECLARATION

```
property Quality: integer;
```

DESCRIPTION

This parameter controls the compression quality for the main image. A value of 1 sets lossless mode. Increasing values result in higher compression ratios and lower image quality. The default value is 1. UseCodecOptions must be true. You have to set this property before each PutFrame call.

WicWriter

Unit IEWic

TIEWICWriter

DECLARATION

```
property Subsampling: integer;
```

DESCRIPTION

This parameter only applies to RGB images. It enables additional compression in the chroma space, preserving luminance detail at the expense of color detail:

Value	Description
3	4:4:4 encoding preserves full chroma resolution.
2	4:2:2 encoding reduces chroma resolution to ½ of luminance resolution.
1	4:2:0 encoding reduces chroma resolution to ¼ of luminance resolution.
0	4:0:0 encoding discards all chroma content, preserving luminance only. Because the codec uses a slightly modified definition of luminance to improve performance, it is preferred to convert an RGB image to monochrome before encoding rather than use this chroma subsampling mode.

Any value greater than 3 returns an error. The default value is 3. UseCodecOptions must be true. You have to set this property before each PutFrame call.

WicWriter

Unit IEWic
TIEWICWriter

DECLARATION

```
property UseCodecOptions: Boolean;
```

DESCRIPTION

If this parameter is true, the Quality, Overlap and Subsampling parameters are used in place of the ImageQuality encoder canonical parameter. When false, the Quality, Overlap and Subsampling parameters are set based on a table lookup determined by the ImageQuality parameter. The default value is false. You have to set this property before each PutFrame call.

Unit IEWic
TIEWICWriter

DECLARATION

```
property VerticalTileSlices: integer;
```

DESCRIPTION

HorizontalTileSlices and VerticalTileSlices specify the horizontal and vertical tiling of the image prior to compression encoding for the most optimal region decode performance. Dividing the image into rectangular tiles during encoding makes it possible to decode regions of the image without the need to process the entire compressed data stream. The default value of 0 specifies no subdivision, so the entire image is treated as a single tile.

A value of 1 for each parameter will create a single horizontal and a single vertical division, effectively dividing the image into four equally sized tiles. The maximum value of 4095 for each parameter divides the image into 4096 tile rows with 4096 tiles per row. In other words, the parameter values equal the number of horizontal and vertical tiles (respectively) minus 1.

A tile can never be smaller than 16 pixels in width or height, so the HD Photo encoder may adjust this parameter to maintain the required minimum tile size.

Because there is storage and processing overhead associated with each tile, these values should be chosen carefully to meet the specific scenario and unless there is a very specific reason, large numbers of small tiles should be avoided.

The default value for both parameters is 0. You have to set this property before each PutFrame call.

WicWriter

Chapter 12

TIERFBClient

Chapter 12. IERFBClient

DESCRIPTION

TIERFBClient implements a RFB (Remote Frame Buffer) client.

It can connect to any RFB compatible server like RealVNC, TightVNC, VMWare virtual machines or Macintosh remote desktop.

Currently implemented features:

Protocol	3.3, 3.7 and 3.8
Authentication	No authentication or VNC (DES) authentication
Pixel format	8 bit with RGB palette, 16 bit RGB, 32 bit RGB
Client messages	SetPixelFormat, SetEncodings, FrameBufferUpdateRequest, KeyEvent, PointerEvent, ClientCutText

Server messages	FrameBufferUpdate, SetColorMapEntries, Bell, ServerCutText
Keyboard	Limited support (CTRL-?, ALT-? could require more code by application)
Encodings	Raw, CopyRect, RRE, Cursor, DesktopSize
Cursor	Cursor shape and drawing local handled

TIERFBClient requires at least Windows 2000.
Keysending doesn't support all key combinations (like CTRL-?, ALT-?, etc...), so applications should handle these combination manually.

See capture\RFB\VNCViewer1 and capture\RFB\VNCViewer2 examples.

Methods and properties

Create

Connect/Disconnect

Connect Connected Disconnect

Suspended

Unit HYIEUtils
TIERFBClient

Server Properties

ScreenName	ScreenPixelFormat	ScreenSize

Commands

SendClipboard	SendKeyEvent	SendPointerEvent
SendRequestUpdate		

Framebuffer Access (to access frame buffer or cursor)

LockFrameBuffer	UnlockFrameBuffer

Frame Buffer And Cursor Bitmaps

Cursor	FrameBuffer

Events

OnBell	OnClipboardText	OnCursorShapeUpdated
OnUpdate	OnUpdateRect	OnUpdateScreenSize

Methods and properties

DECLARATION

```
constructor Create(FrameBuffer: TIEBitmap = nil);
```

DESCRIPTION

Create creates a new instance of TIERFBClient using the specified bitmap as framebuffer. If FrameBuffer is nil then a new bitmap is allocated and owned by the TIERFBClient object.

Connect/disconnect

DECLARATION

```
procedure Connect(const Address: string; Port: word = 5900; const Password: AnsiString = '');
```

DESCRIPTION

Connect connects to the specified RFB (VNC) server. Connect may throw EIERFBError exception if an error occurs (wrong address, wrong port, unsupported protocol version, authentication failure, etc.).

Parameter	Description
Address	IP4, IP6 or name of the RFB server.
Port	TCP Port of the RFB server.
Password	Optional password. Max 8 characters.

Unit HYIEUtils

TIERFBClient

DECLARATION

```
property Connected: Boolean;
```

DESCRIPTION

Connected returns true when connection is active.

See also

Connect

Disconnect

DECLARATION

```
procedure Disconnect();
```

DESCRIPTION

Disconnect disconnects from the server. Waits for message handler thread terminates, so this function could require some time.

IERFBClient

DECLARATION

```
property Suspended: Boolean;
```

DESCRIPTION

Setting this property you can suspend frame buffer updates. This is useful when applications need to do some processing on the framebuffer (like save it TIERFBClient continues to receive messages and updates from the server) when suspended, but they do not update the frame buffer. Events like OnUpdate and OnUpdateRect are disabled when connection is in suspended state.

Server properties

DECLARATION

```
property ScreenName: AnsiString;
```

DESCRIPTION

Returns the server screen name.

DECLARATION

```
property ScreenPixelFormat: TIERFBPixelFormat;
```

DESCRIPTION

Specifies connection pixel format (this is not the framebuffer pixelformat, which is always 24 bit RGB). Applications must set this property before start the connection. Default is ierfbRGB32.

DECLARATION

```
property ScreenSize: TSize;
```

DESCRIPTION

Returns the server screen size (width and height in pixels). This may change after OnUpdateScreenSize event.

Commands

DECLARATION

```
procedure SendClipboard(const Text: AnsiString);
```

DESCRIPTION

SendClipboard sends text to the server clipboard. Only ISO 8859-1 (Latin-1) text is supported by the protocol. SendClipboard may throw EIERFBError exception if a communication error occurs.

DECLARATION

```
procedure SendKeyEvent(xkey: dword; down:
Boolean);
procedure SendKeyEvent(VirtualKey: dword;
KeyData: dword; down: Boolean);
```

DESCRIPTION

Sends a Windows Virtual Key or a XWindow key to the server. The first overload (XWindow) is fully supported (you can send any XWindow key). The second overload (VirtualKey) doesn't support all key combinations (like CTRL-?, ALT-?, etc...), so applications should handle these combination manually. SendKeyEvent may throw EIERFBError exception if a communication error occurs.

Parameter	Description
xkey	XWindow key code.
down	True when the key is down, False otherwise.
VirtualKey	Windows Virtual Key code.
KeyData	Windows Virtual Key data.

Unit HYIEUtils
TIERFBClient

DECLARATION

```
procedure SendPointerEvent(x, y: integer;
LeftButton: Boolean; MiddleButton: Boolean;
RightButton: Boolean);
```

DESCRIPTION

SendPointerEvent communicates a new mouse pointer position and mouse buttons state to the server. SendPointerEvent also updates the frame buffer drawing the mouse cursor, so you should refresh the screen just after.

SendPointerEvent may throw EIERFBError exception if a communication error occurs.

Parameter	Description
x	Horizontal coordinate of the mouse (hotspot), relative to the frame buffer.
y	Vertical coordinate of the mouse (hotspot), relative to the frame buffer..
LeftButton	True if the left button is down.
MiddleButton	True if the middle button is down.
RightButton	True if the right button is down.

IERFBClient

Unit HYIEUtils
TIERFBClient

DECLARATION

```
procedure SendRequestUpdate(x, y, width, height:
word; incremental: Boolean);
procedure SendRequestUpdate(incremental: Boolean
= true);
```

DESCRIPTION

SendRequestUpdate sends an update request to the server for the specified rectangle or for the whole frame buffer. SendRequestUpdate may throw EIERFBError exception if a communication error occurs.

Update requests are sent automatically by TIERFBClient, so under normal circumstances, it is not necessary to call this method.

Parameter	Description
x	Horizontal position of the rectangle to update.
y	Vertical position of the rectangle to update.
width	Width of the rectangle to update.
height	Height of the rectangle to update.
incremental	If true only we need only changes occurred since last update.

Framebuffer access (to access frame buffer or cursor)

DECLARATION

```
procedure LockFrameBuffer;
```

DESCRIPTION

Locks frame buffer and cursor bitmap preventing writing operations. This method could freeze message handler thread so make sure to call UnlockFrameBuffer and to maintain frame buffer locked for less time as possible. Applications could prevent frame buffer updates just setting Suspended = true, which does not freeze the connection.

DECLARATION

```
procedure UnlockFrameBuffer;
```

DESCRIPTION

Unlocks frame buffer and cursor bitmap, locked by LockFrameBuffer. Applications could prevent frame buffer updates just setting Suspended = true, which does not freeze the connection.

IERFBClient

Frame buffer and cursor bitmaps

```
property Cursor: TIEBitmap;
```

DESCRIPTION

Cursor contains last cursor shape sent by the server.

See also
LockFrameBuffer
UnlockFrameBuffer

DECLARATION

```
property FrameBuffer: TIEBitmap;
```

DESCRIPTION

Contains the bitmap where TIERFBClient will paint server screen. This bitmap could be external (for instance a TImageEnView.IEBitmap) or internal (created and owned by TIERFBClient object).

See also
Connect
LockFrameBuffer
UnlockFrameBuffer
Suspended

Events

DECLARATION

```
property OnBell: TNotifyEvent;
```

DESCRIPTION

This event occurs whenever server sends “bell” message.

DECLARATION

```
property OnClipboardText:
TIERFBClipboardTextEvent;
```

DESCRIPTION

This event occurs whenever server sends clipboard text (user on the server Copies or Cuts some text).

DECLARATION

```
property OnCursorShapeUpdated: TNotifyEvent;
```

DESCRIPTION

OnCursorShapeUpdates occurs whenever the cursor shape (Cursor) is updated. Cursor is painted automatically, so you don’t need to handle this event or to read Cursor bitmap.

IERFBClient

TIERFBClient

DECLARATION

```
property OnUpdate: TNotifyEvent;
```

DESCRIPTION

OnUpdate occurs after a sequence of rectangles has been updated (see OnUpdateRect.

DECLARATION

```
property OnUpdateRect: TIERFBUpdateRectEvent;
```

DESCRIPTION

OnUpdateRect occurs whenever server updates a single frame buffer rectangle. Multiple rectangle updates can be sent in a single message so OnUpdate is also sent after a sequence of OnUpdateRect messages.

DECLARATION

```
property OnUpdateScreenSize: TNotifyEvent;
```

DESCRIPTION

OnUpdateScreenSize occurs whenever server changes desktop size. The framebuffer will be automatically resized.

Chapter 13

TIEImageList

Chapter 13. IEImageList

TIEImageList is a simple in memory list of TIEBitmap images. TIEImageList is a class and is not a component. It is very similar to TImageList, except it is non-visual and can hold any image supported by ImageEnIO of any dimensions.

Be aware however that because TIEImageList is a memory list so it is probably not wise to hold many large images in an in-memory list.

TIEImageList could easily be over looked by developers because it is a recent addition to ImageEn and because it is not a component. TIEImageList is an excellent tool to use especially with small images like glyphs, small bitmaps or icons. It can often replace a much harder to use TPageControl that contains a TImageEnView on each page. So for glyph or icon viewer or editor type of applications it will be a valuable addition to your vcl toolkit.

Demos
viewers\manualFlow

Methods and Properties

AppendImageRef	Clear	Filename
FillFromDirectory	Image	ImageCount
Remove		

Methods and Properties

DECLARATION

```
function AppendImageRef(image: TIEBitmap;
filename: WideString): integer;
```

DESCRIPTION

AppendImageRef appends the specified image. This method doesn't copy the bitmap, but just takes ownership of the image object.

Parameter	Description
image	Image to append.
filename	Filename of the image.

Unit HYIEUtils
TIEImageList

DECLARATION

```
procedure Clear();
```

DESCRIPTION

Clear removes all images.

DECLARATION

```
property Filename[index: integer]: WideString;
```

DESCRIPTION

Filename represents the filename of the specified image.

IEImageList

Unit HYIEUtils
TIEImageList

DECLARATION

```
procedure FillFromDirectory(const Directory:
WideString; Limit: integer=-1;
AllowUnknownFormats: Boolean=false; const
ExcludeExtensions: WideString='';
DetectFileFormat: Boolean=false; const
FilterMask: WideString='');
```

DESCRIPTION

FillFromDirectory automatically loads all known images inside Directory.

Parameter	Description
Directory	Directory where to search files.
Limit	Specifies the maximum number of images to load. -1 means no limit.
AllowUnknownFormats	If false (default) loads only known and supported file formats. Otherwise tries to load all files.
ExcludeExtensions	Contains a comma separated list of file extensions to discard (i.e. 'lyr,all,iev').
DetectFileFormat	If true then the image type is detected reading the header, otherwise ImageEn looks at filename extension.
FilterMask	Contains a comma separated list of file extensions to include. Empty string means "all supported extensions".

IEImageList

Unit HYIEUtils
TIEImageList

DECLARATION

```
property Image[index: integer]: TIEBitmap;
```

DESCRIPTION

Image gets/sets the specified image index.

DECLARATION

```
property ImageCount: integer;
```

DESCRIPTION

ImageCount returns number of images in the list.

Read-only.

DECLARATION

```
procedure Remove(imageIndex: integer);
```

DESCRIPTION

Remove removes the specified image.

Chapter 14

TIETIFFHandler

Chapter 14. IETIFFHandler

TIETIFFHandler class allows you to read all TIFF tags, edit/add tags and pages, delete pages, and finally save back the modified TIFF.

This class is unable to work with some OLD-JPEG tiff files. OLD-JPEG compression is no more in TIFF standard.

Demo
Inputoutput / tiffhandler

Unit HYIEUtils
TIETIFFHandler

Methods and Properties

Create

Input/Output

ReadFile	ReadStream	InsertTIFFStream
InsertTIFFFile	InsertPageAsFile	InsertPageAsStream
InsertPageAsImage	WriteFile	WriteStream
FreeData		

Informational

LittleEndian	Version

Tag Info/Handling

GetTagsCount	GetPagesCount	GetTagCode
GetTagType	GetTagLength	GetTagLengthInBytes
FindTag	DeleteTag	GetTagDescription
ChangeTagCode		

Tag Read

GetInteger	GetString	GetFloat
GetValue	SaveTagToFile	GetValueRAW

Tag Write

SetValue	SetValueRAW

Tag Copy

CopyTag

Pages Handling

DeletePage	InsertPage
MovePage	ExchangePage

Methods and Properties

DECLARATION

```
constructor Create(const FileName: string);
constructor Create(Stream: TStream);
```

DESCRIPTION

Create creates TIETIFFHandler object reading image data from file or stream.

Input/Output

DECLARATION

```
function ReadFile(const FileName: string):
Boolean;
```

DESCRIPTION

ReadFile reads the TIFF from file.

IETIFFHandler

DECLARATION

```
function ReadStream(Stream: TStream): Boolean;
```

DESCRIPTION

ReadStream reads the TIFF from stream.

DECLARATION

```
function InsertTIFFStream(Stream: TStream;
pageIndex: integer): Boolean;
```

DESCRIPTION

InsertTIFFStream inserts a TIFF (even if multipage file) at specified page index. The first page has pageIndex=0.

DECLARATION

```
function InsertTIFFFile(const FileName: string;
pageIndex: integer): Boolean;
```

DESCRIPTION

InsertTIFFFile inserts a TIFF (even if multipage file) at specified page index. The first page has pageIndex=0.

IETIFFHandler

DECLARATION

```
function InsertPageAsFile(const FileName:string;
pageIndex: integer):Boolean;
```

DESCRIPTION

Inserts a supported file format (jpeg, bmp, ..) at the specified page index.

DECLARATION

```
function InsertPageAsStream(Stream: TStream;
pageIndex: integer): Boolean;
```

DESCRIPTION

Inserts a supported file format (jpeg, bmp, ..) at the specified page index.

DECLARATION

```
function InsertPageAsImage(viewer: TObject;
pageIndex: integer): Boolean;
```

DESCRIPTION

InsertPageAsImage inserts a loaded image at the specified index. Viewer must be a TImageEnView object or TImageEnIO.

DECLARATION

```
procedure WriteFile(const FileName:string;
pageIndex: integer = -1);
```

DESCRIPTION

WriteFile saves back the modified TIFF. page specifies the page index to write. -1 = all pages.

DECLARATION

```
procedure WriteStream(Stream: TStream; page:
integer = -1);
```

DESCRIPTION

WriteStream saves back the modified TIFF. page specifies the page index to write. -1 = all pages.

DECLARATION

```
procedure FreeData;
```

DESCRIPTION

Frees memory used by the object.

Informational

DECLARATION

```
property LittleEndian: Boolean;
```

DESCRIPTION

Returns true if file has little-endian byte order (Intel byte order). LittleEndian returns false when it has big-endian (Motorola, PowerPC byte order). It is not possible to change byte order.

DECLARATION

```
property Version: word;
```

DESCRIPTION

Returns file version. TIFF ver.6 will have $2A, while Microsoft HD Photo will have $1BC.

IETIFFHandler

Tag Info/Handling

DECLARATION

```
function GetTagsCount(pageIndex: integer): integer;
```

DESCRIPTION

Gets the number of tags presents at specified page.

DECLARATION

```
function GetPagesCount: integer;
```

DESCRIPTION

GetPagesCount gets the number of pages.

DECLARATION

```
function GetTagCode(pageIndex: integer; tagIndex: Integer): integer;
```

DESCRIPTION

Gets the tag code (also named 'tag-id') indexed by tagIndex at specified page. Applications obtain tagIndex calling FindTag.

DECLARATION

```
function GetTagType(pageIndex: integer; tagIndex:
integer): TIETagType;
```

DESCRIPTION

GetTagType gets the tag type of a tag. Applications obtain tagIndex calling FindTag.

DECLARATION

```
function GetTagLength(pageIndex: integer;
tagIndex: integer):integer;
```

DESCRIPTION

GetTagLength gets the tag length of a tag. Look at tag type to know the size of each item. Applications obtain tagIndex calling FindTag.

DECLARATION

```
function GetTagLengthInBytes(pageIndex: integer;
tagIndex: integer):integer;
```

DESCRIPTION

GetTagLengthInBytes gets the tag length in bytes. Applications obtain tagIndex calling FindTag.

IETIFFHandler

DECLARATION

```
function FindTag(pageIndex: integer; tagCode:
integer):integer;
```

DESCRIPTION

FindTag returns the tag index of the first tag which has specified tagCode. If no tag is found, it returns -1.

DECLARATION

```
procedure DeleteTag(pageIndex: integer; tagIndex:
integer);
```

DESCRIPTION

DeleteTag deletes specified tag. Applications obtain tagIndex calling FindTag.

DECLARATION

```
function GetTagDescription(pageIndex: integer;
tagIndex: integer):AnsiString;
```

DESCRIPTION

GetTagDescription returns a short description of specified tag. Applications obtain tagIndex calling FindTag.

IETIFFHandler

Unit HYIEUtils
TIETIFFHandler

DECLARATION

```
procedure ChangeTagCode(pageIndex: integer;
tagIndex: integer; newCode: integer);
```

DESCRIPTION

ChangeTagCode changes the tag code of an existing tag. Applications obtain tagIndex calling FindTag.

Tag Read

DECLARATION

```
function GetInteger(pageIndex: integer; tagIndex:
integer; arrayIndex: integer): integer;
```

DESCRIPTION

GetInteger returns the tag value as Integer. arrayIndex is used only if the tag contains an array of values, otherwise it must be 0. Applications obtain tagIndex calling FindTag.

IETIFFHandler

DECLARATION

```
function GetString(pageIndex: integer; tagIndex: integer): AnsiString;
```

DESCRIPTION

GetString returns the tag value as string. Applications obtain tagIndex calling FindTag.

DECLARATION

```
function GetFloat(pageIndex: integer; tagIndex: integer; arrayIndex: integer): double;
```

DESCRIPTION

GetFloat returns the tag value as floating point value. arrayIndex is used only if the tag contains an array of values, otherwise it must be 0. Applications obtain tagIndex calling FindTag.

DECLARATION

```
function GetValue(pageIndex: integer; tagIndex: integer; arrayIndex: integer):variant;
```

DESCRIPTION

GetValue returns the tag value as variant. ArrayIndex is used only if the tag contains an array of values, otherwise it must be 0.
Applications obtain tagIndex calling FindTag.

IETIFFHandler

DECLARATION

```
procedure SaveTagToFile(pageIndex: integer;
tagIndex: integer; const fileName: string);
```

DESCRIPTION

SaveTagToFile saves the content of a tag to file.
Applications obtain tagIndex calling FindTag.

DECLARATION

```
function GetValueRAW(pageIndex: integer;
tagIndex: integer; arrayIndex: integer): pointer;
```

DESCRIPTION

GetValueRAW returns a pointer to the tag value. arrayIndex is used only if the tag contains an array of values, otherwise it must be 0. Use GetTagLength or GetTagLengthInBytes to know the size of buffer.
The returned buffer must not be freed. Applications obtain tagIndex calling FindTag.

IETIFFHandler

Tag Write

DECLARATION

```
procedure SetValue(pageIndex: integer; tagCode:
integer; tagType: TIETagType; value: variant);
```

DESCRIPTION

Set the value of a tag. If the tag doesn't exist a new one is created.

DECLARATION

```
procedure SetValueRAW(pageIndex: integer;
tagCode: integer; tagType: TIETagType; dataNum:
integer; buffer: pointer);
```

DESCRIPTION

SetValueRAW sets the value of tag as raw buffer. If the tag doesn't exist a new one is created. DataNum is the number of items of type tagType in the buffer. This is not the buffer length in bytes.

IETIFFHandler

Tag Copy

DECLARATION

```
procedure CopyTag(srcPageIndex: integer;
srcTagIndex: integer; source: TIETIFFHandler;
dstPageIndex: integer);
```

DESCRIPTION

Copies a tag from another TIETIFFHandler object (that is, from another TIFF). Applications obtain srcTagIndex calling FindTag.

Parameter	Description
srcPageIndex	Source page index (0=first page)
srcTagIndex	Source tag index (use FindTag to get an index from tag code)
source	Source TIETIFFHandler object, that is the source TIFF file
dstPageIndex	Destination page index (0=first page)

Page Handling

DECLARATION

```
procedure DeletePage(pageIndex: integer);
```

DESCRIPTION

DeletePage deletes the specified page.

DECLARATION

```
function InsertPage(pageIndex:  integer = -1):
integer;
function InsertPage(pageIndex:  integer;
sourceHandler: TIETIFFHandler; sourcePage:
integer): integer;
```

DESCRIPTION

The first overload inserts a new blank page at specified index. The second overload inserts sourcePage from sourceHandler at specified index. If pageIndex is -1 or equal to GetPagesCount the page is added at end. InsertPage returns the index of added page.

DECLARATION

```
procedure MovePage(CurIndex, NewIndex: integer);
```

DESCRIPTION

Moves specified page to a new position.

DECLARATION

```
procedure ExchangePage(Index1, Index2:integer);
```

DESCRIPTION

Exchanges specified pages.

IETIFFHandler

TIETagsHandler

The TIETagsHandler class allows reading EXIF maker note tags, when it is in form of IFD (like Canon).

Methods and Properties

GetInteger	GetIntegerArray	GetIntegerIndexed
GetMiniString	GetRational	GetRationalIndexed
GetRawData	GetString	TagExists
TagLength		

Methods and Properties

DECLARATION

```
function GetInteger(Tag: integer): integer;
```

DESCRIPTION

GetInteger returns the specified tag as integer.

IETagsHandler

DECLARATION

```
function GetIntegerArray(Tag: integer; var ar:
PDWordArray): integer;
```

DESCRIPTION

GetIntegerArray returns the specified tag as an array of integers.

DECLARATION

```
function GetIntegerIndexed(Tag: integer; index:
integer): integer;
```

DESCRIPTION

GetIntegerIndexed returns the specified tag as an array of integers.

DECLARATION

```
function GetMiniString(Tag: integer): AnsiString;
```

DESCRIPTION

GetMiniString returns the specified tag as a string of 4 bytes.

IETagsHandler

DECLARATION

```
function GetRational(Tag: integer; defaultValue: double = 0): double;
```

DESCRIPTION

GetRational returns the specified tag as double.

DECLARATION

```
function GetRationalIndexed(Tag: integer; index: integer; defVal: double = 0.0): double;
```

DESCRIPTION

GetRationalIndexed returns the specified tag as an array of doubles.

DECLARATION

```
function GetRawData(Tag: integer): pointer;
```

DESCRIPTION

GetRawData returns the specified tag as a raw bytes buffer.

IETagsHandler

DECLARATION

```
function GetString(Tag: integer): AnsiString;
```

DESCRIPTION

GetString returns the specified tag as string.

DECLARATION

```
function TagExists(Tag: integer): Boolean;
```

DESCRIPTION

TagExists returns true if the specified tag exists.

DECLARATION

```
function TagLength(Tag: integer): integer;
```

DESCRIPTION

TagLength returns the tag length in byte.

Chapter 15

IEResourceExtractor

Chapter 15. IEResourceExtractor

TIEResourceExtractor class extracts image resources from a PE files like EXE, DLL, OCX, ICL, BPL, CPL, etc.

Look at inputoutput/resourceLoader example or see ImageEnResourceExtractor.zip for a Delphi 2010 project which is similar to the inputoutput/resourceLoader example with many more features. The ImageEnResourceExtractor requires the TShellTreeView, and TShellListView which is distributed with Delphi but not automatically installed. The ImageEnResourceExtractor demonstrates all of the TIEResourceExtractor helper functions (IEResourceExtractorHelperFunctions.pas) which are a part of this book.

Example

```
// loads resource 143, in "Bitmap"s, from
// "explorer.exe" (should be a little Windows
//logo)
var
re: TIEResourceExtractor;
buffer: pointer;
bufferLen: integer;
begin
  re :=
    TIEResourceExtractor.Create('explorer.exe');
  try
    buffer := re.GetBuffer('Bitmap',
      'INTRESOURCE:143', bufferLen);
    ImageEnView1.IO.Params.IsResource := true;
    ImageEnView1.IO.LoadFromBuffer(buffer,
       bufferLen, ioBMP);
  finally
    re.Free;
  end;
end;
```

Unit HYIEUtils
TIEResourceExtractor

Methods and properties

Create	FriendlyTypes	GetBuffer
GetFrameBuffer	GetGroupAndFrame	GetResourceBookmark
GroupCountFrames	GroupFrameDepth	GroupFrameHeight
GroupFrameName	GroupFrameWidth	IndexOfType
IsGroup	IsGrouped	IsValid
Names	NamesCount	TypesCount
Types	Helper Functions	

Methods and properties

DECLARATION

```
constructor Create(const Filename: WideString);
```

DESCRIPTION

Create creates a new instance of TIEResourceExtractor loading the specified PE file (EXE, DLL, OCX, ICL, BPL, etc.). It is possible to check the success of loading reading IsValid property. FileName specifies the path and filename of PE module.

DECLARATION

```
property FriendlyTypes[TypeIndex: integer]:
AnsiString;
```

DESCRIPTION

FriendlyTypes returns the specified resource type friendly name (for known types like RT_CURSOR, RT_BITMAP, etc.).

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type, TypesCount-1 is last resource type.

Look also Types for not so friendly type strings.

IEResourceExtractor

DECLARATION

```
function GetBuffer(TypeIndex: integer; NameIndex:
integer; var BufferLength: integer): pointer;
function GetBuffer(const TypeStr: AnsiString;
const NameStr: AnsiString; var BufferLength:
integer): pointer;
function GetBuffer(ResourceBookmark:
TIEResourceBookmark; var BufferLength: integer):
pointer;
```

DESCRIPTION

GetBuffer returns memory buffer for the specified resource.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type; TypesCount-1 is last resource type.
NameIndex	Index of actual resource. 0 is first resource name; NamesCount-1 is last resource name.
BufferLength	Field filled with the resulting buffer length (in bytes).
TypeStr	Type as string (ie 'Bitmap', 'Cursor').
NameStr	Resource name as string (ie 'INTRESOURCE:100', 'Hand').
ResourceBookmark	A resource bookmark returned by GetResourceBookmark.

The buffer must not be freed.

DECLARATION

```
function GetFrameBuffer(TypeIndex: integer;
NameIndex: integer; FrameIndex: integer; var
BufferLength: integer): pointer;
```

DESCRIPTION

GetFrameBuffer returns the buffer of specified frame, for multi-frame resources. TypeIndex must be 'GroupIcon' or 'GroupCursor'.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type; TypesCount-1 is last resource type.
NameIndex	Index of actual resource. 0 is first resource name; NamesCount-1 is last resource name.
FrameIndex	The frame index. 0 is first resource name, GroupCountFrames-1 is last frame.
BufferLength	

IEResourceExtractor

Unit HYIEUtils
TIEResourceExtractor

DECLARATION

```
procedure GetGroupAndFrame(TypeIndex: integer;
NameIndex: integer; var GroupTypeIndex: integer;
var GroupIndex: integer; var GroupFrameIndex:
integer);
```

DESCRIPTION

This method finds the associated grouping resource for the specified resource. Grouping resources types are “GroupIcon” or “GroupCursor”. TypeIndex must be “Icon” or “Cursor”, otherwise returns values are undefined.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type; TypesCount-1 is last resource type.
NameIndex	Index of actual resource. 0 is first resource name; NamesCount-1 is last resource name.
GroupTypeIndex	<Return value: associated group resource type./C>
GroupIndex	<Return value: associated group resource name./C>
GroupFrameIndex	<Return value: associated group resource frame./C>

IEResourceExtractor

Unit HYIEUtils
TIEResourceExtractor

DECLARATION

```
function GetResourceBookmark(TypeIndex: integer;
NameIndex: integer; FrameIndex: integer = -1):
TIEResourceBookmark;
```

DESCRIPTION

GetResourceBookmark creates a bookmark for the specified resource (or resource frame). Bookmarks are automatically freed.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type; TypesCount-1 is last resource type.
NameIndex	Index of actual resource. 0 is first resource name; NamesCount-1 is last resource name.
FrameIndex	The frame index. 0 is first resource name; GroupCountFrames-1 is last frame.

IEResourceExtractor

Unit HYIEUtils
TIEResourceExtractor

DECLARATION

```
property GroupCountFrames[TypeIndex: integer;
NameIndex: integer]: integer;
```

DESCRIPTION

GroupCountFrames returns the number of frames of specified resource (can be Icon or Cursor resource).

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type; TypesCount-1 is last resource type.
NameIndex	Index of actual resource. 0 is first resource name; NamesCount-1 is last resource name.

Unit HYIEUtils
TIEResourceExtractor

DECLARATION

```
property GroupFrameDepth[TypeIndex: integer;
NameIndex: integer; FrameIndex: integer]:
integer;
```

DESCRIPTION

GroupFrameDepth returns the icon or cursor bit depth. TypeIndex must refer to a “GroupIcon” or “GroupCursor” resource type.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type; TypesCount-1 is last resource type.
NameIndex	Index of actual resource. 0 is first resource name; NamesCount-1 is last resource name.

IEResourceExtractor

Unit HYIEUtils
TIEResourceExtractor

DECLARATION

```
property GroupFrameHeight[TypeIndex: integer;
NameIndex: integer; FrameIndex: integer]:
integer;
```

DESCRIPTION

GroupFrameHeight returns the icon or cursor height. TypeIndex must refer to a “GroupIcon” or “GroupCursor” resource type.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type; TypesCount-1 is last resource type.
NameIndex	Index of actual resource. 0 is first resource name; NamesCount-1 is last resource name.

Unit HYIEUtils
TIEResourceExtractor

DECLARATION

```
property GroupFrameName[TypeIndex: integer;
NameIndex: integer; FrameIndex: integer]:
AnsiString;
```

DESCRIPTION

GroupFrameName returns the icon or cursor resource name. TypeIndex must refer to a “GroupIcon” or “GroupCursor” resource type.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type; TypesCount-1 is last resource type.
NameIndex	Index of actual resource. 0 is first resource name; NamesCount-1 is last resource name.

DECLARATION

```
property GroupFrameWidth[TypeIndex: integer;
NameIndex: integer; FrameIndex: integer]:
integer;
```

DESCRIPTION

GroupFrameWidth returns the icon or cursor width. TypeIndex must refer to a “GroupIcon” or “GroupCursor” resource type.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type, TypesCount-1 is last resource type.
NameIndex	Index of actual resource. 0 is first resource name; NamesCount-1 is last resource name.

DECLARATION

```
function IndexOfType(TypeName: AnsiString):
integer;
```

DESCRIPTION

IndexOfType finds the index of specified type name (ex. ‘Icon’, ‘Cursor’, ‘GroupIcon’...).

DECLARATION

```
property IsGroup[TypeIndex: integer]: Boolean;
```

DESCRIPTION

IsGroup returns true if the resource type is “GroupIcon” (RT_GROUP_ICON) or “GroupCursor” (RT_GROUP_CURSOR). This means that resources inside this type must be handled as groups.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type; TypesCount-1 is last resource type.

DECLARATION

```
property IsGrouped[TypeIndex: integer]: Boolean;
```

DESCRIPTION

Returns true if the resource type is “Icon” (RT_ICON) or “Cursor” (RT_CURSOR). This means that resources inside this type must be grouped in “GroupIcon” or “GroupCursor” resources.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type, TypesCount-1 is last resource type.

IEResourceExtractor

Unit HYIEUtils
TIEResourceExtractor

DECLARATION

```
property IsValid: Boolean;
```

DESCRIPTION

IsValid checks if TIEResourceExtractor contains valid data.

DECLARATION

```
property Names[TypeIndex: integer; NameIndex:
integer]: AnsiString;
```

DESCRIPTION

Names return the resource name for specified type and name index.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type, TypesCount-1 is last resource type.
NameIndex	Index of actual resource. 0 is first resource name, NamesCount-1 is last resource name.

IEResourceExtractor

DECLARATION

```
property NamesCount[TypeIndex: integer]: integer;
```

DESCRIPTION

NamesCount returns number of resource names found in the PE module, for the specified resource type.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type, TypesCount-1 is last resource type.

DECLARATION

```
property TypesCount: integer;
```

DESCRIPTION

TypesCount returns number of resource types found in the PE module.

IEResourceExtractor

DECLARATION

```
property Types[TypeIndex: integer]: AnsiString;
```

DESCRIPTION

Types return the specified resource type name.

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type, TypesCount-1 is last resource type.

Look also FriendlyTypes for more friendly type strings.

DECLARATION

```
property FriendlyTypes[TypeIndex: integer]:
AnsiString;
```

DESCRIPTION

FriendlyTypes returns the specified resource type friendly name (for known types like RT_CURSOR, RT_BITMAP, etc...).

Parameter	Description
TypeIndex	Index of resource type. 0 is first resource type, TypesCount-1 is last resource type.

Look also Types for not so friendly type strings.

IEResourceExtractor

Helper Functions For TIEResourceExtractor

The TIEResourceExtractor is lacking a number of useful functions to make it a bit easier to use. We have written these functions which are not a part of ImageEn itself:

```
function NumberOfResourcesInRes( APath: string ):
integer;
// Return the number of resources in the resource
//  file
var
  i: integer;
  j: integer;
  iTotalResources: integer;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := 0;
  iTotalResources := 0;
  iResourceExtractor := TIEResourceExtractor.Create(
     APath );
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount-1
      do
        for j := 0 to
           iResourceExtractor.NamesCount[i] - 1
           do
              if iResourceExtractor.IsGroup[i]
              then
               inc(iTotalResources,
             iResourceExtractor.GroupCountFrames[
             i, j])
```

```
          else
            inc( iTotalResources );
      Result := iTotalResources;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;

function NumberOfImagesInRes( APath: string ):
integer;
// Return the number of images in the resource file
var
  i: integer;
  j: integer;
  iTotalImages: integer;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := 0;
  iTotalImages := 0;
  iResourceExtractor :=
     TIEResourceExtractor.Create(APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount
       - 1 do
        for j := 0 to
           iResourceExtractor.NamesCount[i] - 1
           do
           if ( iResourceExtractor.IsGroup[i] )
           and ( (
               ResourceExtractor.FriendlyTypes[i
                ] = 'GroupIcon' ) or
              (iResourceExtractor.FriendlyTypes[
                 i] = 'Image') or
              (iResourceExtractor.FriendlyTypes[
                i] = 'Icon' ) or
```

```
                (iResourceExtractor.FriendlyTypes[
                  i] = 'GroupCursor' ) or
                (iResourceExtractor.FriendlyTypes[
                  i] = 'Cursor' ) or
                (iResourceExtractor.FriendlyTypes[
                 i] = 'Bitmap' ) or
                (iResourceExtractor.FriendlyTypes[
                 i] = 'PNG' ) ) then
              inc(iTotalImages,
               iResourceExtractor.GroupCountFrames[
               i, j]) else if (
                  iResourceExtractor.FriendlyTypes[
                     i] = 'Image') or
               (iResourceExtractor.FriendlyTypes[
                     i] = 'Icon' ) or
               (iResourceExtractor.FriendlyTypes[
                     i] = 'Cursor' ) or
               (iResourceExtractor.FriendlyTypes[i] =
                  'Bitmap') or
               (iResourceExtractor.FriendlyTypes[i] =
               'PNG') then
                   inc(iTotalImages);
              Result := iTotalImages;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

```
function NumberOfGroupIconsInRes(APath: string):
integer;
// Return the number of groupicons in the resource
// file
var
  i: integer;
  j: integer;
  iTotalImages: integer;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := 0;
  iTotalImages := 0;
  iResourceExtractor :=
     TIEResourceExtractor.Create( APath );
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount
         - 1 do
        for j := 0 to
           iResourceExtractor.NamesCount[i] - 1
             do
          if ( iResourceExtractor.IsGroup[i] )
            and ((
              iResourceExtractor.FriendlyTypes[i
              ] = 'GroupIcon')) then
            inc(iTotalImages);
      Result := iTotalImages;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

Unit HYIEUtils
TIEResourceExtractor

```
function NumberOfGroupCursorsInRes(APath: string):
integer;
// Return the number of groupcursors in the resource
file
var
  i: integer;
  j: integer;
  iTotalImages: integer;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := 0;
  iTotalImages := 0;
  iResourceExtractor :=
     TIEResourceExtractor.Create(APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
        do
        for j := 0 to
           iResourceExtractor.NamesCount[i] - 1
          do
            if (iResourceExtractor.IsGroup[i]) and ((
                iResourceExtractor.FriendlyTypes[
               i] = 'GroupCursor')) then
                inc(iTotalImages,
             iResourceExtractor.GroupCountFrames[
             i, j]);
      Result := iTotalImages;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

Unit HYIEUtils
TIEResourceExtractor

```
function NumberOfBMPInRes(APath: string): integer;
// Return the number of bmp in the resource file
var
  i: integer;
  j: integer;
  iTotalImages: integer;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := 0;
  iTotalImages := 0;
  iResourceExtractor := TIEResourceExtractor.Create(
APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
      do
      for j := 0 to
        iResourceExtractor.NamesCount[i] - 1 do
          if iResourceExtractor.FriendlyTypes[i
             ] = 'Bitmap' then
            inc(iTotalImages);
      Result := iTotalImages;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

Unit HYIEUtils
TIEResourceExtractor

```
function NumberOfIconsInRes(APath: string): integer;
// Return the number of ico in the resource file
var
  i: integer;
  j: integer;
  iTotalImages: integer;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := 0;
  iTotalImages := 0;
  iResourceExtractor :=
  TIEResourceExtractor.Create(APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
         do
        for j := 0 to
          iResourceExtractor.NamesCount[i] - 1
        do
          if iResourceExtractor.FriendlyTypes[i
             ] = 'Icon' then
            inc(iTotalImages);
      Result := iTotalImages;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

Unit HYIEUtils
TIEResourceExtractor

```
function NumberOfCursorsInRes(APath: string):
integer;
// Return the number of cur in the resource file
var
  i: integer;
  j: integer;
  iTotalImages: integer;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := 0;
  iTotalImages := 0;
  iResourceExtractor :=
     TIEResourceExtractor.Create(APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount
         - 1 do
        for j := 0 to
          iResourceExtractor.NamesCount[i] - 1
        do
          if iResourceExtractor.FriendlyTypes[i
             ] = 'Cursor' then
            inc(iTotalImages);
          Result := iTotalImages;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

Unit HYIEUtils
TIEResourceExtractor

```
function NumberOfBitmapsInRes(APath: string):
integer;
// Return the number of bmp in the resource file
var
  i: integer;
  j: integer;
  iTotalImages: integer;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := 0;
  iTotalImages := 0;
  iResourceExtractor := TIEResourceExtractor.Create(
     APath);
  try
    if iResourceExtractor.IsValid then
    begin

      for i := 0 to iResourceExtractor.TypesCount
          - 1 do
        for j := 0 to
           iResourceExtractor.NamesCount[i] - 1
        do
          if iResourceExtractor.FriendlyTypes[i
             ] = 'Bitmap' then
            inc(iTotalImages);
      Result := iTotalImages;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

```
function NumberOfPNGInRes(APath: string): integer;
// Return the number of png in the resource file
var
  i: integer;
  j: integer;
  iTotalImages: integer;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := -1;
  iTotalImages := 0;
  iResourceExtractor := TIEResourceExtractor.Create(
     APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
        do
        for j := 0 to iResourceExtractor.NamesCount[
           i ] - 1 do
          if iResourceExtractor.FriendlyTypes[i] =
             'PNG' then
          inc(iTotalImages);
      Result := iTotalImages;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

```
function NumberOfExtractableImagesInRes(APath: string
): integer;
// Return the number of extractable images (bmp, png,
// groupicons and icons) in the resource file
var
  iNumberOfExtractableImagesInRes: integer;
  iNumberOfBitmaps: integer;
  iNumberOfPNG: integer;
  iNumberOfGroupIcons: integer;
  iNumberOfIcons: integer;
begin
  iNumberOfBitmaps := NumberOfBitmapsInRes(APath);
  iNumberOfPNG := NumberOfPNGInRes(APath);
  iNumberOfGroupIcons := NumberOfGroupIconsInRes(
APath);
  iNumberOfIcons := NumberOfIconsInRes(APath);
  iNumberOfExtractableImagesInRes:= iNumberOfBitmaps
+  NumberOfPNG + iNumberOfGroupIcons +
   iNumberOfIcons;
  Result := iNumberOfExtractableImagesInRes;
end;

function ResourceHasBMP(APath: string): boolean;
// Return true if the resource has bitmaps, false if
not
begin
  Result := NumberOfBMPInRes(APath) > 0;
end;

function ResourceHasPNG(APath: string): boolean;
// Return true if the resource has png, false if //
not
begin
  Result := NumberOfPNGInRes(APath) > 0;
end;
```

```
function ResourceHasIcons(APath: string): boolean;
// Return true if the resource has ico, false if
// not
begin
  Result := NumberOfIconsInRes(APath) > 0;
end;

function ResourceHasGroupIcons(APath: string):
boolean;
// Return true if the resource has groupicons,
// false if not
begin
  Result := NumberOfGroupIconsInRes(APath) > 0;
end;

function ResourceHasImages(APath: string): boolean;
// Return true if the resource has images, false if
// not
begin
  Result := NumberOfImagesInRes(APath) > 0;
end;
```

Unit HYIEUtils
TIEResourceExtractor

```
function DoesResourceContainGroupIcon(APath: string
): boolean;
// Returns True if the resource file contains a
// GroupIcon, or false of not
var
  i: integer;
  iFriendlyType: string;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := False;
  iResourceExtractor := TIEResourceExtractor.Create(
     APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
      do
      begin
        iFriendlyType := string(
           iResourceExtractor.FriendlyTypes[i]);
        Result := iFriendlyType = 'GroupIcon';
        if Result then
          Break;
      end;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

Unit HYIEUtils
TIEResourceExtractor

```
function DoesResourceContainIcon(APath: string):
boolean;
// Returns True if the resource file contains a Icon,
or false of not
var
  i: integer;
  iFriendlyType: string;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := False;
  iResourceExtractor := TIEResourceExtractor.Create(
    APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
      do
      begin
        iFriendlyType := string(
           iResourceExtractor.FriendlyTypes[i]);
        Result := iFriendlyType = 'Icon';
        if Result then
          Break;
      end;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

Unit HYIEUtils
TIEResourceExtractor

```
function DoesResourceContainCursor( APath: string ):
boolean;
// Returns True if the resource file contains a
// Cursor, or false of not
var
  i: integer;
  iFriendlyType: string;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := False;
  iResourceExtractor := TIEResourceExtractor.Create(
     APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
      do
      begin
        iFriendlyType := string(
        iResourceExtractor.FriendlyTypes[i]);
        Result := iFriendlyType = 'Cursor';
        if Result then
          Break;
      end;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

Unit HYIEUtils
TIEResourceExtractor

```
function DoesResourceContainGroupCursor(APath: string
): boolean;
// Returns True if the resource file contains a
// Cursor, or false of not
var
  i: integer;
  iFriendlyType: string;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := False;
  iResourceExtractor := TIEResourceExtractor.Create(
     APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
      do
      begin
        iFriendlyType := string(
           iResourceExtractor.FriendlyTypes[i]);
        Result := iFriendlyType = 'GroupCursor';
        if Result then
          Break;
      end;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

Unit HYIEUtils
TIEResourceExtractor

```
function DoesResourceContainBitmap(APath: string):
boolean;
// Returns True if the resource file contains a
Bitmap, or false of not
var
  i: integer;
  iFriendlyType: string;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := False;
  iResourceExtractor := TIEResourceExtractor.Create(
     APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
      do
      begin
        iFriendlyType := string(
           iResourceExtractor.FriendlyTypes[i]);
        Result := iFriendlyType = 'Bitmap';
        if Result then
          Break;
      end;
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

Unit HYIEUtils
TIEResourceExtractor

```
function DoesResourceContainPNG(APath: string):
boolean;
// Returns True if the resource file contains a PNG,
// or false of no
var
  i: integer;
  iFriendlyType: string;
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := False;
  iResourceExtractor := TIEResourceExtractor.Create(
    APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
      do
      begin
        iFriendlyType := string(
           iResourceExtractor.FriendlyTypes[i]);
        Result := iFriendlyType = 'PNG';
        if Result then
          Break;
      end;
    end;
  finally
     iResourceExtractor.Free;
  end;
end;
```

Unit HYIEUtils
TIEResourceExtractor

```
function IsResourceGroupIcon(APath: string; AIndex:
integer ): boolean;
// Returns True if AIndex is a GroupIcon resource, or
// false of not
var
  iIndex: integer;
  iResourceExtractor: TIEResourceExtractor;
  iFriendlyType: string;
  iIsGroup: boolean;
  iIsGrouped: boolean;
begin
  Result := False;
  iResourceExtractor := TIEResourceExtractor.Create(
     APath);
  try
    if iResourceExtractor.IsValid then
    begin
      // search for GroupIcon
      iIndex := AIndex;
      iFriendlyType := string(
         iResourceExtractor.FriendlyTypes[iIndex]);
      iIsGroup := iResourceExtractor.IsGroup[iIndex];
      iIsGrouped := iResourceExtractor.IsGrouped[
         iIndex];
      if (iFriendlyType = 'GroupIcon') or (iIsGroup)
         or (iIsGrouped) then
        Result := True
      else
        Result := False;
    end;
  finally
    iResourceExtractor.Free();
  end;
end;
```

```
function IsResourceIcon(APath: string; ASelection:
string; AIndex: integer): boolean;
// Returns True if ASelection is a Icon resource, or
// false of not
var
  i: integer;
  j: integer;
  iFriendlyType: string;
  iResourceExtractor: TIEResourceExtractor;
  iResourceExtractorName: string;
begin
  Result := False;
  iResourceExtractor := TIEResourceExtractor.Create(
     APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
      do
      begin
        iFriendlyType := string(
        iResourceExtractor.FriendlyTypes[AIndex]);
        if i <> 0 then
          if iFriendlyType = string(
            iResourceExtractor.FriendlyTypes[i])
             then
                Break;
        for j := 0 to iResourceExtractor.NamesCount[
           i] - 1 do
        begin
          // Use the selection to select the resource
          iResourceExtractorName := string(
             iResourceExtractor.Names[i, j]);
          if iResourceExtractorName = ASelection then
          begin
            iFriendlyType := string(
               iResourceExtractor.FriendlyTypes[i]);
            Break;
```

```
          end;
        end;
      end;
      Result := iFriendlyType = 'Icon';
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

```
function IsResourceGroupCursor(APath: string; AIndex:
integer): boolean;
// Returns True if ASelection is a GroupCursor
// resource, or false of not
var
  iResourceExtractor: TIEResourceExtractor;
begin
  Result := False;
  iResourceExtractor := TIEResourceExtractor.Create(
    APath);
  try
    if iResourceExtractor.IsValid then
    begin
      // search for GroupCursor
      if iResourceExtractor.FriendlyTypes[AIndex] =
         'GroupCursor' then
        Result := True
      else
        Result := False;
    end;
  finally
     iResourceExtractor.Free();
  end;
end;
```

```
function IsResourceCursor(APath: string; ASelection:
string): boolean;
// Returns True if ASelection is a Cursor resource,
// or false of not
var
  i: integer;
  j: integer;
  iFriendlyType: string;
  iResourceExtractor: TIEResourceExtractor;
  iResourceExtractorName: string;
begin
  Result := False;
  iResourceExtractor := TIEResourceExtractor.Create(
     APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
      do
      begin
        if i <> 0 then
          if iFriendlyType = string(
             iResourceExtractor.FriendlyTypes[i - 1]
           then
              Break;
        for j := 0 to iResourceExtractor.NamesCount[
           i ] - 1 do
        begin
          // Use the selection to select the resource
          iResourceExtractorName := string(
          iResourceExtractor.Names[i, j]);
          if iResourceExtractorName = ASelection then
          begin
            iFriendlyType := string(
            iResourceExtractor.FriendlyTypes[i]);
```

```
            Break;
          end;
        end;
      end;
      Result := iFriendlyType = 'Cursor';
    end;
  finally
    iResourceExtractor.Free;
  end;
end;

function IsResourceBitmap(APath: string; ASelection:
string): boolean;
// Returns True if ASelection is a Bitmap resource,
// or false of not
var
  i: integer;
  j: integer;
  iFriendlyType: string;
  iResourceExtractor: TIEResourceExtractor;
  iResourceExtractorName: string;
begin
  Result := False;
  iResourceExtractor := TIEResourceExtractor.Create(
     APath);
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
      do
      begin
        if i <> 0 then
          if iFriendlyType = string(
             iResourceExtractor.FriendlyTypes[i - 1])
             then
                Break;
```

```
        for j := 0 to iResourceExtractor.NamesCount[
          i] - 1 do
        begin
          // Use the selection to select the resource
          iResourceExtractorName := string(
          iResourceExtractor.Names[i, j]);
          if iResourceExtractorName = ASelection then
          begin
            iFriendlyType := string(
               iResourceExtractor.FriendlyTypes[i]);
            Break;
          end;
        end;
      end;
      Result := iFriendlyType = 'Bitmap';
    end;
  finally
    iResourceExtractor.Free;
  end;
end;

function IsResourcePNG(APath: string; ASelection:
string): boolean;
// Returns True if ASelection is a PNG resource, or
// false of not
var
  i: integer;
  j: integer;
  iFriendlyType: string;
  iResourceExtractor: TIEResourceExtractor;
  iResourceExtractorName: string;
begin
  Result := False;
  iResourceExtractor := TIEResourceExtractor.Create(
     APath);
```

```
  try
    if iResourceExtractor.IsValid then
    begin
      for i := 0 to iResourceExtractor.TypesCount - 1
      do
      begin
        if i <> 0 then
          if iFriendlyType = string(
             iResourceExtractor.FriendlyTypes[i - 1])
             then
                 Break;
        for j := 0 to iResourceExtractor.NamesCount[
           i] - 1 do
        begin
          // Use the selection to select the resource
          iResourceExtractorName := string(
             iResourceExtractor.Names[i, j]);
          if iResourceExtractorName = ASelection then
          begin
            iFriendlyType := string(
               iResourceExtractor.FriendlyTypes[i]);
            Break;
          end;
        end;
      end;
      Result := iFriendlyType = 'PNG';
    end;
  finally
    iResourceExtractor.Free;
  end;
end;
```

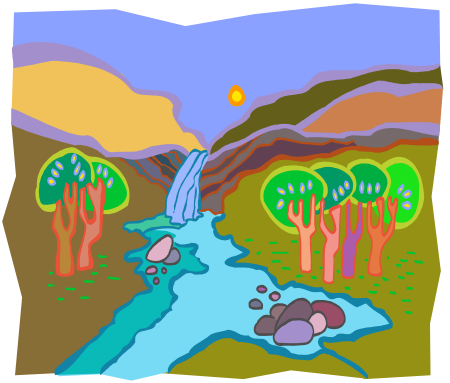
Unit HYIEUtils
TIEResourceExtractor

Chapter 16

IEHashStream

Chapter 16. IEHashStream

TIEHashStream builds a hash string from a stream.



Hash algorithms can be MD2, MD4, MD5 and SHA.

TIEHashStream creates a TIEHashStream which will use specified hash algorithm.

If Buffered is true the stream data is written in a temporary memory stream. This is necessary when Seek and Read methods are necessary.

- TIEDICOMTags
- TIEICC

TIECC class contains a color profile. It is used for the loaded image ICC profile and for the display ICC profile (which default is sRGB).

Note: several constants are defined in ielcms unit.

Examples

```
// saves the file with an unique name (create
// hash from the jpeg content and use it as file
// name)
var
hashStream: TIEHashStream;
begin
  hashStream := TIEHashStream.Create(iehaMD5);
  try
    ImageEnView1.IO.SaveToStreamJpeg(hashStream);
      hashStream.SaveToFile(hashStream.GetHash()+
        '.jpg');
  finally
    hashStream.Free;
  end;
end;
```

Methods and properties

Create	GetHash	LoadFromFile
LoadFromStream	Write	Read
Seek	SaveToFile	SaveToStream

Methods and properties

DECLARATION

```
constructor Create(Algorithm: TIEHashAlgorithm =
iehaMD5; Buffered: Boolean = true);
```

DESCRIPTION

Create creates a TIEHashStream which will use specified hash algorithm. If Buffered is true the stream data is written in a temporary memory stream. This is necessary when Seek and Read methods are necessary.

DECLARATION

```
function GetHash: AnsiString;
```

DESCRIPTION

Calculates the hash and returns the string representation of the hash.

DECLARATION

```
procedure LoadFromFile(const filename:
AnsiString);
```

DESCRIPTION

LoadFromFile loads data to hash from specified file.
It is necessary to create the stream as “buffered” to use this method.

IEHashStream

DECLARATION

```
procedure LoadFromStream(Stream: TStream);
```

DESCRIPTION

LoadFromStream loads data to hash from specified stream.
It is necessary to create the stream as “buffered” to use this method.

DECLARATION

```
function Write(const Buffer; Count: longint):
longint;
```

DESCRIPTION

Write writes data in the hash stream.

DECLARATION

```
function Read(var Buffer; Count: longint):
Longint;
```

DESCRIPTION

Read reads data in the hash stream. It is necessary to create the stream as “buffered” to use this method.

IEHashStream

Unit HYIEUtils
TIEHashStream

DECLARATION

```
function Seek(const Offset:int64; Origin:
TSeekOrigin): int64;
```

DESCRIPTION

Seek seeks data in the hash stream. It is necessary to create the stream as “buffered” to use this method.

DECLARATION

```
procedure SaveToFile(const filename: AnsiString);
```

DESCRIPTION

Saves the stream to file (this is the actual data written, not the hash). This is useful to save the hashed data one time. It is necessary to create the stream as “buffered” to use this method.

See also
SaveToStream.

IEHashStream

Unit HYIEUtils
TIEHashStream

DECLARATION

```
procedure SaveToStream(Stream: TStream);
```

DESCRIPTION

Saves data to stream (this is the actual data written, not the hash). This is useful to save the hashed data one time. It is necessary to create the stream as “buffered” to use this method.

See also
SaveToFile

IEHashStream

Chapter 17

Image Processing

Chapter 17. ImageEnProc

TImageEnProc provides image processing and analysis tasks on attached TImageEnView, TImageEnDBView, TImageEn, TIEBitmap, TImage or TBitmap components.

Also, it handles all clipboard operations. It includes an area selection capability for most of image processing tasks (only if the attached component isn't TImageEnView, which already has the area selection capability).

TImageEnView already encapsulates TImageEnIO and TImageEnProc, so you don't actually need to put a TImageEnProc on the form.

Removing RedEye

There may be other ways, but try selecting an eye with miSelectCircle, after you make the selection, position the selection over the first eye... Then **press and hold the SHIFT Key** and select the second eye. Unfortunately I know of no way to position the second selection precisely over the eye without moving the selection over the first eye.

If you happen to click on the screen after making both selections, both selections will be removed. If you try to reposition the second selection over the second eye, the first selection will move with the second selection and will probably not be where you want it to be (over the first eye), so you have to click on the screen to clear the selection and start over.

It would be nice to be able to position multiple selections around on the screen for this sort of thing, so you could select the first eye, reposition the selection precisely where you want it on the first eye, then **press and hold the SHIFT key** to select the second eye, then reposition the selection precisely where you want it over the second eye but ImageEn does not support this feature.

The good news is even if you do not have the selections precisely where you want, but the selection covers all of the red part of the eye then you call RemoveRedEyes the area around the eye is not changed very much because the selection is quite small and only the portion of the selection that is red will be changed.

What this means is that precise selection of just the red area of an eye is usually not necessary, but if your selection does not include all of the red part of the eye, the unselected red part of the eye will not be changed. In this case you will have to reselect the missed part of the redeye and try again.

Unit ImageENProc

TImageEnProc

Methods and Properties

Connected Component

AttachedBitmap	AttachedIEBitmap	AttachedImageEn
AttachedTImage		

Dialogs

DoPreviews	IPDialogParams	MsgLanguage
PreviewFont	PreviewsLog	PreviewsParams

Shadows

AddInnerShadow	AddSoftShadow

Fourier Analysis (FFT)

FTClearZone	FTConvertFrom	FTCreateImage
FTDisplayFrom		

Paint

CastColor	ClearSel	Clear
Fill	PaintMark	PaintPenMarker

Unit ImageENProc
TImageEnProc

Alpha Channel

CastAlpha	SetTransparentColors

Custom Image Processing

BeginImageProcessing	EndImageProcessing

Custom Image Analysis

BeginImageAnalysis	EndImageAnalysis

Analysis

CalcAverageRGB	CalcDensityHistogram
CalcImageNumColors	CalcImagePalette
CalcOrientation	CompareWith
CompareHistogramWith	ComputeImageEquality
GetDominantColor	GetHistogram
GetHSVChannelAll	GetHSVChannel
GetRGBChannelAll	GetRGBChannel
SeparateObjects	

Unit ImageENProc
TImageEnProc

Noise

RemoveIsolatedPixels RemoveNoise

Pixels

AdjustBrightnessContrast	AdjustLumSatHistogram	AdjustSaturation
AdjustTemperature	AdjustTint	ApplyFilterPreset
ApplyFilter	Blur	BumpMapping
CastColorRange	Closing	Colorize
Contrast2	Contrast3	Contrast
ConvertTo24Bit	ConvertToBWFloydSteinberg	ConvertToBWLocalThreshold
ConvertToBWOrdered	ConvertToBWThreshold	ConvertToGray
ConvertToPalette	ConvertTo	Convolve
DisposeChannels	Deinterlace	EdgeDetect_ShenCasta
EdgeDetect_Sobel	FiltersInitialDir	GammaCorrect
HistEqualize	HSLvar	HSVvar
IntensityRGBAll	Intensity	Lens
MapGrayToColor	Maximum	MedianFilter
MergeIEBitmap	Merge	MatchHSVRange
Minimum	MotionBlur	Negative
Opening	Random	RemoveRedEyes
Sharpen	Threshold	Threshold2

UnsharpMask	WallisFilter	Wave
WhiteBalance_coef	WhiteBalance_WhiteAt	

Clipboard

CopyToClipboard	IsClipboardAvailable	PasteFromClipboard
PointPasteFromClip	SelCopyToClip	SelCutToClip
SelPasteFromClipStretch	SelPasteFromClip	

Steganography

ClearHiddenText	GetHiddenDataSpace	ReadHiddenData
ReadHiddenText	WriteHiddenData	WriteHiddenText

Encrypting

Decrypt	Encrypt

Geometric

AutoCrop	AutoCrop2	CropSel
Crop	Flip	ImageResize
MakeTile	PerspectiveDraw	ProjectDraw
RadialStretch	Reflection	Resample
ResampleTo	RotateAndCrop	Rotate

Unit ImageENProc
TImageEnProc

RoundImage ShiftChannel SkewDetectionFine

SkewDetection

Automatic Image Enhancement

AdjustGainOffset AutoImageEnhance1 AutoImageEnhance2

AutoImageEnhance3 AutoSharp HistAutoEqualize

WhiteBalance_AutoWhite WhiteBalance_GrayWorld

Undo

AutoUndo CanUndo ClearAllUndo

ClearUndoAt ClearUndo SaveUndoCaptioned

SaveUndo UndoAt UndoCaptions

UndoCount UndoLimit UndoLocation

UndoPeekAt UndoRect Undo

Redo

CanRedo ClearAllRedo ClearRedo

RedoAt RedoCaptions RedoCount

RedoPeekAt Redo SaveRedoCapti

SaveRedo

Unit ImageENProc
TImageEnProc

Transitions

PrepareTransitionBitmaps PrepareTransitionBitmapsEx

CreateTransitionBitmap

Others

AutoConvertFormat Background Create

CreateFromBitmap GetReSel Update

Events

OnFinishWork OnPreview OnProgress

OnSaveUndo

Methods and Properties

Connected component

DECLARATION

```
property AttachedBitmap: TBitmap;
```

DESCRIPTION

Use this property if you want to attach TImageEnProc to a normal TBitmap object. This property is mutually exclusive with AttachedTImage and AttachedImageEn.

Unit ImageENProc

TImageEnProc

DECLARATION

```
property AttachedIEBitmap: TIEBitmap;
```

DESCRIPTION

AttachedIEBitmap specifies the attached TIEBitmap object. Using TIEBitmap instead of TBitmap allows TImageEnProc to be thread safe and to handle large images.

DECLARATION

```
property AttachedImageEn: TIEView;
```

DESCRIPTION

Use this property if you want to attach TImageEnProc to a TImageEn, TImageEnView or TImageEnDBView (or any other inherited object) component. This property is mutually exclusive with AttachedTImage and AttachedBitmap.

DECLARATION

```
property AttachedTImage: TImage;
```

DESCRIPTION

Use this property if you want to attach TImageEnProc to a TImage (any other inherited object) component. This property is mutually exclusive with AttachedImageEn and <TImageEnProc.AttachedBitmap>.

Dialogs

Declaration

```
function DoPreviews(pe: TPreviewEffects =
[peAll]; IsResizeable: Boolean=true; FormWidth:
integer=-1; FormHeight: integer=-1; FormLeft:
integer=-1; FormTop: integer=-1): Boolean;
```

Description

This method executes the image processing preview dialog. When IsResizeable is true then user can resize the dialog. FormWidth and FormHeight controls the initial form size. -1 = default value. FormLeft and FormTop controls the initial form position. -1 = default value (centered).

pe is the set of the image effects to show in the dialog.

There are two constants for color adjustment and general effect sets:
ppeColorAdjust
ppeEffects

Declaration

```
property IPDialogParams: TIPDialogParams;
```

Description

This property contains TIPDialogParams object which allows get/set several parameters used to control Image Processing Dialog called by DoPreviews method.

Declaration

```
property MsgLanguage: TMsgLanguage;
```

Description

This property sets the language for the image processing previews dialog (Previews).

Declaration

```
property PreviewFont: TFont;
```

Description

The PreviewFont property contains the font used in the Previews dialog (DoPreviews). Make sure the size of the font matches label's length.

Unit ImageENProc

TImageEnProc

DECLARATION

```
property PreviewsLog: TStringList;
```

DESCRIPTION

Represents a log of operations executed inside preview dialog (DoPreviews).

DECLARATION

```
property PreviewsParams:  TPRPreviewsParams;
```

DESCRIPTION

This property specifies some features that the image processing preview dialog will have.

Unit ImageENProc

TImageEnProc

DECLARATION

TPRPreviewsParams = set of TPRPreviewsParamsItems;
TPRPreviewsParamsItems = (prppDefaultLockPreview, prppShowResetButton, prppHardReset, prppResetSelectedTab);

DESCRIPTION

PreviewsParams specifies some features that the image processing preview dialog will have.

Value	Description
prppDefaultLockPreview	Set down the “Lock preview” button when the dialog is shown.
prppShowResetButton	Show a “Reset” button, which resets parameters to previous values.
prppHardReset	Allow Reset button to reset to default values instead of previous values.
prppResetSelectedTab	Allow Reset button to reset only selected tab instead of all tabs.

Shadows

Declaration

```
procedure AddInnerShadow(radius: double; OffSetX: integer; OffSetY: integer; ShadowColor: TColor = clBlack);
```

Description

AddInnserShadow creates a shadow in the inner border of the image.

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure AddSoftShadow(radius: double; OffSetX: integer; OffSetY: integer; AdaptSize: Boolean; ShadowColor: TColor; Intensity:integer);
```

DESCRIPTION

AddSoftShadow adds a soft shadow (Gaussian Shadow) to the image. AddSoftShadow creates the shadow effect using the image's alpha channel (creates it if it doesn't exist). To see the effect, set EnableAlphaChannel=true. Radius specifies the shadow wide. OffsetX and OffsetY specify the shadow offset from the image. If AdaptSize is true (the default) the image is resized to contain the shadow. ShadowColor (default black) is the shadow color. Intensity (default 100) is the shadow intensity, with values in the range 0 and 100 (maximum).

Fourier analysis (FFT)

DECLARATION

```
procedure FTClearZone(tx1, ty1, tx2, ty2:integer;
GrayScale: Boolean);
```

DESCRIPTION

Parameter	Description
tx1	Left coordinate of reduced FFT map. Corresponds to IPDialogParams.FFT_Left.
ty1	Top coordinate of reduced FFT map. Corresponds to IPDialogParams.FFT_Top.
tx2	Right coordinate of reduced FFT map. Corresponds to IPDialogParams.FFT_Right.
ty2	Bottom coordinate of reduced FFT map. Corresponds to IPDialogParams.FFT_Bottom.
GrayScale	Makes the bitmap grayscale. Corresponds to IPDialogParams.FFT_GrayScale.

Unit ImageENProc
TImageEnProc

DECLARATION

```
procedure FTConvertFrom(ft: TIEftImage);
```

DESCRIPTION

FTConvertFrom performs an inverse Fourier Transformation of ft (Fourier transformed image). ft is created by FTCreateImage method.

DECLARATION

```
function FTCreateImage(ImageType: TIEFtImageType;
NewWidth, NewHeight: integer): TIEFtImage;
```

DESCRIPTION

FTCreateImage creates an object that contains the Fourier transformation of the current image.

ImageType specifies the kind of the output transformation and can be ieitRGB (to build a RGB Fourier transformation) or ieitGrayscale (to build a gray scaled Fourier transformation). NewWidth and NewHeight are useful to resample the current image before applying the Fourier transformation (to speed up the job). Set -1 to get original values. TIEFtImage is a Delphi object and can be destroyed with the Free method.

To access to the complex data of the transformed image you have to use the TIEFtImage object that is declared in the iefft unit.

TIEFtImage exports ComplexPixel[], ComplexWidth and ComplexHeight properties.ComplexPixel[] is defined as: ComplexPixel[x,y:integer]:TIEComplexColor; ComplexPixel[] returns the complex pixel at x,y coordinate. TIEComplexColor is defined as:
TIEComplexColor=packed record
// red channel
real_Red:PIEsingle;
imag_Red:PIEsingle;
// blue channel
real_Blue:PIEsingle;
imag_Blue:PIEsingle;
// green channel
real_Green:PIEsingle;
mag_Green:PIEsingle;
// gray scale
imag_gray:PIEsingle;
real_gray:PIEsingle;
end;

The fields ???_Red, ???_Blue and ???_Green are filled if ImageType is ieitRGB. The fields ???_gray are filled if ImageType is ieitGrayscale. ComplexWidth and ComplexHeight are the width and the height of complex image.

For example, to set complex pixel 0,0 to 0.1 write (for ImageType=ieitRGB):

```
ftimage.ComplexPixel[0,0].real_Red^:=0.1;
ftimage.ComplexPixel[0,0].imag_Red^:=0.1;
ftimage.ComplexPixel[0,0].real_Green^:=0.1;
ftimage.ComplexPixel[0,0].imag_Green^:=0.1;
ftimage.ComplexPixel[0,0].real_Blue^:=0.1;
ftimage.ComplexPixel[0,0].imag_Blue^:=0.1;
```

DECLARATION

```
procedure FTDisplayFrom(ft: TIEftImage);
```

DESCRIPTION

FTDisplayFrom builds an image that is the “visible” representation of Fourier transformation ft. Ft is created by FTCreateImage method.

Paint

DECLARATION

```
procedure CastColor(x, y:integer; newColor: TRGB;
tolerance: integer);
```

DESCRIPTION

CastColor performs a flood-fill starting at x, y coordinates. NewColor is the color used to paint. Tolerance specifies the maximum difference from the starting pixel (0=no tolerance, pixels must be equals).

DECLARATION

```
procedure ClearSel;
```

DESCRIPTION

ClearSel fills the selected region with the current background color.

Unit ImageENProc

TImageEnProc

Declaration

```
procedure Clear;
```

Description

Use this method to completely fill the current image with the Background color.

Declaration

```
procedure Fill(FillColor: TRGB);
procedure Fill(FillColor: TColor);
```

Description

Fill sets all colors of the selected area with FillColor color. For black/white images FillColor can be only (0,0,0) for black and (255,255,255) for white.

Declaration

```
procedure PaintMark(Frequency: integer; Color:
TRGB);
```

Description

PaintMark paints a pixel of Color every Frequency pixels, over the selected area.

Unit ImageENProc

TImageEnProc

Declaration

```
procedure PaintPenMarker(x,y: integer; Width:
integer=20; Color: TColor=clYellow;
BackgroundColor: TColor=clWhite; Tolerance:
integer=10);
```

Description

PaintPenMarker emulates a pen marker over the image (that must be in true color).

X and y specify the position of the pen in bitmap coordinates. Width specifies the size of the circle. BackgroundColor is the background color to change, replaced with Color. Tolerance is the maximum difference from BackgroundColor to allow drawing the marker.

PaintPenMarker doesn't save the previous image into an Undo buffer.

Alpha Channel

Declaration

```
procedure CastAlpha(x, y:integer; newAlpha:
integer; tolerance: integer);
```

Description

Performs a flood-fill starting at x,y coordinates. NewAlpha is the transparent value used to fill the area. Tolerance specifies the maximum difference from the starting pixel (0=no tolerance, pixels must be equals).

Declaration

```
procedure SetTransparentColors(MinColor,MaxColor:
TRGB; Alpha: integer);
```

Description

SetTransparentColors sets as transparent all pixels inside the MinColor and MaxColor range. The transparency value comes from Alpha parameter (0=fully transparent, 255=fully visible). If you are sure of the RGB value of the transparent colors, just set MinColor and MaxColor to the same value. If you want all colors from gray (128,128,128) to white (255,255,255) to be transparent, write SetTransparentColors(CreateRGB(128,128,128), CreateRGB(255,255,255),0).

Unit ImageENProc

TImageEnProc

Custom Image Processing

DECLARATION

```
function BeginImageProcessing(allowedFormats:
TIEPixelFormatSet; var x1,y1,x2,y2: integer;
const OpName: string; var ProcBitmap: TIEBitmap;
var mask: TIEMask): Boolean;
```

DESCRIPTION

BeginImageProcessing and <TImageEnProc.EndImageProcessing> allow you to create custom image processing functions handling automatically selection area, pixel format consistency and undo.

AllowedFormats specifies the pixel formats allowed.
x1, y1, x2, y2 are the destination rectangle coordinates where to apply the function. OpName is a string describing the function. ProcBitmap is the processing bitmap. Mask is the selection mask.

Using BeginImageProcessing/EndImageProcessing you can avoid considering if the selection is rectangle, elliptical, irregular or magic wand, just process the bitmap as a rectangle.

ImageEnProc

Unit ImageENProc

TImageEnProc

Declaration

```
procedure EndImageProcessing(ProcBitmap:
TIEBitmap; mask: TIEMask);
```

Description

BeginImageProcessing and EndImageProcessing allow you to create custom image processing functions handling automatically selection area, pixel format consistency and undo.

ProcBitmap is the processing bitmap coming from <TImageEnProc.BeginImageProcessing>.

Mask is the selection mask coming from <TImageEnProc.BeginImageProcessing>.

Using BeginImageProcessing/EndImageProcessing you can avoid considering if the selection is rectangle, elliptical, irregular or magic wand, just process the bitmap as a rectangle.

Declaration

```
function BeginImageAnalysis(allowedFormats:
TIEPixelFormatSet; var x1, y1, x2, y2: integer;
var ProcBitmap: TIEBitmap; var mask: TIEMask):
boolean;
```

Description

BeginImageAnalysis and EndImageAnalysis allow you to create custom image analysis functions handling automatically selection area, pixel format consistency.

ImageEnProc

AllowedFormats specifies the pixel formats allowed. x1, y1, x2, y2 are the destination rectangle coordinates where to apply the function. ProcBitmap is the processing bitmap. Mask is the selection mask.

Using BeginImageAnalysis/EndImageAnalysis you can avoid considering if the selection is rectangle, elliptical, irregular or magicwand, just process the bitmap as a rectangle.

Example

```
procedure SearchWhitePixel( Proc: TImageEnProc );
var
  ProcBitmap: TIEBitmap;
  Mask: TIEMask;
  x1, y1, x2, y2:integer;
  x,y: integer;
  px: PRGB;
begin
  // we support only ie24RGB format
  if not  Proc.BeginImageAnalysis([ie24RGB],x1,y1,x2,y2,ProcBitmap,
     mask) then
    exit;
  for y := y1 to y2-1 do begin
    px := ProcBitmap.Scanline[y];
    for x := x1 to x2-1 do begin
      with px^ do
        if (r=255) and (g=255) and (b=255) then
          ShowMessage('Found White Pixel!');
      inc(px);
    end;
  end;
  // finalize
  proc.EndImageAnalysis(ProcBitmap);
end;

ImageEnView.SelectEllipse( 100,100, 100,100 );
SearchWhitePixel( ImageEnView.Proc );
```

Unit ImageENProc

TImageEnProc

Declaration

```
procedure EndImageAnalysis(ProcBitmap: TIEBitmap);
```

Description

BeginImageAnalysis and EndImageAnalysis allow you to create custom image analysis functions handling automatically selection area, pixel format consistency. ProcBitmap is the processing bitmap coming from BeginImageAnalysis.

Using BeginImageAnalysis/EndImageAnalysis you can avoid considering if the selection is rectangle, elliptical, irregular or magicwand, just process the bitmap as a rectangle.

***Example* See**

SearchWhitePixel

Analysis

Declaration

```
function CalcAverageRGB():TRGB;
```

Description

CalcAverageRGB returns the average RGB values inside the selected area.

ImageEnProc

Declaration

```
procedure
CalcDensityHistogram(VertHist,HorizHist: pointer;
norm_vert,norm_horiz: integer);
```

Description

CalcDensityHistogram calculates the pixel vertical and density histograms. See the example 'DensityAnalysis' for more information.

Declaration

```
function CalcImageNumColors: integer;
```

Description

CalcImageNumColors returns how any colors the current image is using. This value doesn't depend on the internal format of the image (so 24 bit images don't necessarily have 16 million colors). The result is the number of found colors in the current image (without limits).

Unit ImageENProc
TImageEnProc

DECLARATION

```
procedure CalcImagePalette(var Palette: array of
TRGB; MaxCol: integer);
```

DESCRIPTION

The CalcImagePalette method fills the Palette array with the colors found in the current image. Palette is the array to fill. MaxCol is the size of Palette. If image has more than MaxCol colors, the image is quantized to match the number of colors specified (MaxCol).

DECLARATION

```
function CalcOrientation: integer;
```

DESCRIPTION

Apply only to document images (images with text) to get their orientation. CalcOrientation can detect only 0° or 90° orientation. It cannot detect upside down, inverted text.

CalcOrientation is useful to adjust a document from portrait to landscape.

Declaration

```
function CompareWith(SecondImage: TIEBitmap;
DiffBitmap: TIEBitmap): double;
```

Description

Compares current image with SecondImage and returns a floating point value from 0 to 1 which specifies the percentage of equality. 1 means that two images are equal. The algorithm compares only the intensity of the pixels, not the colors.

The DiffBitmap can be nil, otherwise it must be an 8 bit bitmap (ie8g or ie8p PixelFormat) which will contain a bitmap with the differences. Images must have same size and have PixelFormat=ie24RGB. See 'motiondetector' example for more info.

Declaration

```
function CompareHistogramWith(SecondImage:
TIEBitmap; Mode: TIECmpMode; GrayScale:
Boolean):double;
```

Description

Compares histograms of current image with SecondImage and returns a floating point value from 0 to 1 which specifies the percentage of equality. Mode specifies the algorithm to compare. If GrayScale is true, only gray scale histogram is compared.

Unit ImageENProc

TImageEnProc

Declaration

```
function ComputeImageEquality(SecondImage:
TIEBitmap; var psnr_min, psnr_max: double; var
mse_min, mse_max: double; var rmse_min, rmse_max:
double; var pae_min, pae_max: double; var
mae_min, mae_max: double):Boolean;
```

Description

ComputeImageEquality calculates some values to find the similitude of current image with the SecondImage.

The method returns true if the images are equal.
Images must have some size and have PixelFormat = ie24RGB.

Returned values:

psnr_min, psnr_max : minimum and maximum peak signal to noise ratio
mse_min, mse_max : minimum and maximum mean squared error
rmse_min, rmse_max : minimum and maximum root mean squared error
pae_min, pae_max : minimum and maximum peak absolute error
mae_min, mae_max : minimum and maximum mean absolute error

Declaration

```
function GetDominantColor(var Color:
TRGB):double;
```

Description

GetDominantColor returns the dominant (most used) color in the image. Color will contain the dominant color, and the returned result will contain the percentage of the dominant color.

Declaration

```
procedure GetHistogram(Hist: pointer);
```

Description

The GetHistogram method fills Hist with the histogram of the current image. Hist is a pointer to THistogram array.

Declaration

```
procedure GetHSVChannelAll(BitmapH, BitmapS,
BitmapV: TIEBitmap);
```

Description

The GetHSVChannelAll method copies the H, S and V channels to BitmapH, BitmapS and BitmapV bitmaps.

Unit ImageENProc

TImageEnProc

DECLARATION

```
function GetHSVChannel(ch: integer): TIEBitmap;
```

DESCRIPTION

The GetHSVChannel function obtains a Bitmap with the HSV specified channel. The result Bitmap is a gray levels representation of specified channel.

Ch is the HSV channel. 0 is Hue, 1 is Saturation, and 2 is Value.

DECLARATION

```
procedure GetRGBChannelAll(BitmapR, BitmapG,
BitmapB: TIEBitmap);
```

DESCRIPTION

GetRGBChannelAll copies the R, G and B channels to BitmapR, BitmapG and BitmapB bitmaps.

DECLARATION

```
function GetRGBChannel(ch: integer): TIEBitmap;
```

DESCRIPTION

GetRGBChannel obtains a Bitmap with the RGB specified channel. The resulting Bitmap is a gray levels representation of specified channel. Ch is the RGB channel. 0 is Blue, 1 is Green and 2 is Red.

Unit ImageENProc

TImageEnProc

DECLARATION

```
function SeparateObjects(Quality: integer=4;
MergeCommonAreas: Boolean=true): TList;
function SeparateObjects(Quality: integer;
MergeCommonAreas: Boolean; BackgroundColorBegin,
BackgroundColorEnd: TRGB):TList; overload;
```

DESCRIPTION

SeparateObjects creates a list (TList) of TRect pointers. Each rectangle encloses found objects. This method works well to separate photos or simple objects over a white or black background. The background cannot contain a pattern.

Quality specifies the contour search routine definition. Minimum value is 1, suggested is 4. Lower values increases speed, but could not recognize complex objects like characters. If MergeCommonAreas is true, when two rectangles intersects they are merged. When it is false, two rectangles are merged only if they are inclusive. BackgroundColorBegin and BackgroundColorEnd specify the background color range. This helps the function to separate the background and the objects.

You have to free the returned list executing:

```
Rects:=ImageEnView1.Proc.SeparateObjects;
..process rects..
// free rects
for i:=0 to rects.Count-1 do
dispose(Prect(rects[i]));
rects.free;
```

Look at imageprocessing1\SeparateObjects example.

ImageEnProc

Noise

Declaration

```
procedure RemoveIsolatedPixels(NoiseColor: integer; PixelsCount: integer);
```

Description

RemoveIsolatedPixels removes all groups of PixelCount isolated pixels. The RemoveIsolatedPixels method works only with black/white images (1bit).

NoiseColor specifies the “text” color, 0 is black and 1 is white (for example if you have a document where the text is black NoiseColor must be 0, otherwise it must be 1).

Declaration

```
procedure RemoveNoise(Iterations: integer; InvertImage: Boolean);
```

Description

RemoveNoise removes noise pixels using the Kfill algorithm. The RemoveNoise method works only with black/white images (1bit).

Iterations are the maximum number of iterations (from 1).
If InvertImage is true, the image is inverted (negative) when the algorithm works.

Pixels

Declaration

```
procedure
AdjustBrightnessContrastSaturation(Brightness,
Contrast, Saturation: integer);
```

Description

This method adjust brightness, contrast and color saturation in one step.

Brightness can assume values from -100 to 512.
Contrast can assume values from -100 to 100.
Saturation can assume values from 0 to 512.

If a parameter is zero, it doesn't change the image.

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure AdjustLumSatHistogram(Saturation,
Luminance: double);
```

DESCRIPTION

AdjustLumSatHistogram adjusts Saturation, Luminance and the histogram.

Saturation allows values from 0 to 1.
Luminance allows values from 0 to 1.

DECLARATION

```
procedure AdjustSaturation(Amount: integer);
```

DESCRIPTION

AdjustSaturation adjusts color saturation. Amount allows values from -100 to 100.

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure AdjustTemperature(temperature:
integer);
```

DESCRIPTION

AdjustTemperature adjusts image temperature. Temperature is measured in degrees Kelvin. Minimum value is 1000 (K) and maximum is 40000 (K). Resolution is 100K (so allowed values are 1000, 1100, 1200, etc.).

DECLARATION

```
procedure AdjustTint(Amount: integer);
```

DESCRIPTION

AdjustTint adjusts color tint. Amount is measured in degrees and allows values from -180 to 180.

Unit ImageENProc

TImageEnProc

Declaration

```
procedure ApplyFilterPreset(filter:
TIEFilterPresets);
```

Description

ApplyFilterPreset applies a preset filter to the current image. TIEFilterPresets is defined as integer and can be one of following constants:
fpNone
fpBlur
fpEdge
fpEmboss
fpHighPass1
fpHighPass2
fpHighPass3
fpLowPass1
fpLowPass2

Declaration

```
procedure ApplyFilter(filter: TGraphFilter);
```

Description

This method applies a 3x3 filter to the current image (or to selected region).

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure Blur(radius: double);
```

DESCRIPTION

Blur performs a Gaussian Blur filter with specified radius (>0).

DECLARATION

```
procedure BumpMapping(LightX, LightY, LampX,
LampY, pcf: integer; Color: TRGB);
```

DESCRIPTION

BumpMapping applies the bump mapping effect to the current image. This effect is available in image processing preview dialog.

LightX and LightY specify source light position.

LampX and LampY specify width and height of the source light.

Pcf is the percentage of the effect to apply to the original image (0..100).

Color is the source light color.

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure CastColorRange(BeginColor, EndColor,
CastColor: TRGB);
```

DESCRIPTION

Use this method to force all colors included in the range BeginColor and EndColor to be equal to CastColor.

DECLARATION

```
procedure Closing(WindowSize: integer);
```

DESCRIPTION

The method Closing performs a dilation (Maximum filter with WindowSize parameter) followed by an erosion (Minimum filter with WindowSize parameter).

The closing filter operation will reduce small negative oriented regions and negative noise regions.

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure Colorize(hue: integer; saturation: integer; luminosity: double);
```

DESCRIPTION

Colorize sets hue and saturation for all pixels of the image. It also changes the luminosity using this parameter as a multiplier to increase or decrease the luminosity.

Parameter	Description
hue	Assumes values between 0 and 359 (corresponding to 0..359 degrees around hexcone).
saturation	Assumes values between 0 (shade of gray) to 99 (pure color).
luminosity	This is 1 when you don't touch the original luminosity.

This function could be used to obtain a sepia effect, using following parameters: ImageEnView.Proc.Colorize(40, 50, 1.1);

ImageEnProc

Unit ImageENProc
TImageEnProc

Declaration

```
procedure Contrast2(Amount: double);
```

Description

This is another contrast adjustment function. Amount assumes floating point values from 0 to 1.

Declaration

```
procedure Contrast3(Change, Midpoint: integer;
DoRed, DoGreen, DoBlue: Boolean);
```

Description

Applies contrast to the image by lightening or darkening pixels depending on whether they are above or below a threshold. The Midpoint value determines that threshold, the point at which the Change value influences the pixel value lighter or darker.

When Change values are positive, increasing the Midpoint value darkens the contrast. When Change values are negative, increasing the Midpoint value lightens the contrast. Midpoint - an integer between -100 and +100 that determines the threshold of light and dark. Change - an integer between -255 and 255 that indicates the level of contrast to be applied each side of the midpoint.

DoRed
DoGreen
DoBlue - If True, applies the change to this RGB channel.

ImageEnProc

DECLARATION

```
procedure Contrast(vv: double);
```

DESCRIPTION

This method changes the contrast of selected region.
Vv is the contrast value which may range from -100 to +100, where 0 is no change.

DECLARATION

```
procedure ConvertTo24Bit;
```

DESCRIPTION

Converts a black & white (pf1bit) image to true color (pf24bit).

DECLARATION

```
procedure ConvertToBW_FloydSteinberg;
```

DESCRIPTION

Converts current image to black & white using FloydSteinberg algorithm

Unit ImageENProc

TImageEnProc

Declaration

```
procedure ConvertToBWLocalThreshold(WinSize:
integer; Mode: TIEThreshMode; Offset: integer);
```

Description

Converts a true color image (24 bit) to black & white (1 bit) using local threshold algorithms (mean, median, min/max mean). WinSize specifies the window size, minimum value is 2. Offset subtracts an offset to the calculated local threshold. Mode specifies the algorithm to use.

Declaration

```
procedure ConvertToBWOrdered;
```

Description

ConvertToBWOrdered converts a true color image (24 bit) to black & white (1 bit) with a dithering ordered method.

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure ConvertToBWThreshold(Threshold:
integer);
```

DESCRIPTION

ConvertToBWThreshold converts a true color image (24 bit) to black & white (1 bit) using a thresholding algorithm.

The image is first converted to gray levels, then all levels that are less of Threshold will be set to black, otherwise will be set to white. Threshold is an intensity value (0..255). If Threshold is -1, the threshold value actually used will be the average of all levels of the original image. If Threshold is -2, a Maximum Entropy Algorithm is used.

DECLARATION

```
procedure ConvertToGray;
```

DESCRIPTION

This method convert's selected region to gray levels. The image always will be in true color (16M of colors). When LegacyBitmap is True, ImageEn can handle only black/white (pf1bit) or true color images (pf24bit).

ConvertToGray just set the R,G and B channels to the same value.

Unit ImageENProc

TImageEnProc

Declaration

```
procedure ConvertToPalette(NumColors: integer;
Palette: pointer; DitherMethod: TIEDitherMethod);
```

Description

ConvertToPalette reduces number of colors to NumColor value and fills the Palette array with the color map used to reduce colors.

DitherMethod specifies how to convert a color image to a black/white. It can be ieOrdered or ieThreshold.

Declaration

```
procedure ConvertTo(NumColors:  integer;
DitherMethod: TIEDitherMethod);
```

Description

The ConvertTo method reduces number of colors to NumColor value. DitherMethod specifies how to convert a color image to a black/white. It can be ieOrdered or ieThreshold.

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure
CreateTransitionBitmap(TransitionProgress :
Single; DestBitmap : TBitmap);
```

DESCRIPTION

Use with PrepareTransitionBitmaps or PrepareTransitionBitmapsEx to create a series of frames that transition from one bitmap to another.

See also

PrepareTransitionBitmaps

PrepareTransitionBitmapsEx

Parameter	Description
TransitionProgress	The percentage that it is has progressed from the start image to the end image (ranging from 0.0 to 100.0)
DestBitmap	Will be filled with the created transition frame. It must be created before calling and will be automatically sized and set to 24 bit

ImageEnProc

Example

```
procedure TransitionFrameCreationExample;
var
OldBitmap, NewBitmap, TransBitmap: TBitmap;
I: Integer;
TransLevel: Single;
begin
  OldBitmap := TBitmap.Create;
  NewBitmap := TBitmap.Create;
  TransBitmap := TBitmap.Create;
  try
    OldBitmap.LoadFromFile('C:\OldImage.bmp');
    NewBitmap.LoadFromFile('C:\NewImage.bmp');
    // Call PrepareTransitionBitmaps once
    ImageEnProc1.PrepareTransitionBitmaps(
    OldBitmap, NewBitmap, iettCrossDissolve);
    for i := 1 to 9 do
    begin
      // Transition levels from 10% to 90%
      TransLevel := i * 10;
      // Call CreateTransitionBitmap for each
      // required frame
      ImageEnProc1.CreateTransitionBitmap(
      TransLevel, TransBitmap);
      TransBitmap.SaveToFile('C:\TransImage' +
       IntToStr(I) + '.bmp');
    end;
  finally
    OldBitmap.Free;
    NewBitmap.Free;
    TransBitmap.Free;
  end;
end;
```

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure Convolve(Kernel: array of double;
KernelWidth, KernelHeight: integer; Factor:
double);
```

DESCRIPTION

Convolve convolves the specified kernel over the selected region.

Parameter	Description
Kernel	The convolution kernel (just a matrix made array).
KernelWidth	The kernel matrix width (number of columns).
KernelHeight	The kernel matrix height (number of rows).
Factor	Multiplication factor.

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure DisposeChannels(newDispo: string);
```

DESCRIPTION

This method changes the channels order from BGR (default) to the specified one.

Parameter	Description
newDispo	A string of three characters, one for each channel ('RGB', 'RBG', etc.). Can contain also '0' to set a channel to zero. You can also replicate the same channel: 'RRR' or 'GGG', etc.

DECLARATION

```
procedure Deinterlace(mode: TIEDeinterlaceMode);
```

DESCRIPTION

This procedure deinterlaces current image with the specified algorithm.

ImageEnProc

Unit ImageENProc
TImageEnProc

DECLARATION

```
procedure EdgeDetect_ShenCastan(Ratio: double;
Smooth: double; WindowSize: integer; ThinFactor:
integer; DoHysteresis: Boolean);
```

DESCRIPTION

EdgeDetect_ShenCastan converts the current color image to black/white (1 bit) with a Shen-Castan (ISEF) edge detection algorithm.

Shen-Castan algorithm convolves the image with the Infinite Symmetric Exponential Filter, computes the binary Laplacian image, suppresses false zero crossing, performs adaptive gradient thresholding, and, finally, also applies hysteresis thresholding. (Algorithms for Image Processing and Computer Vision - J.R.Parker).

Ratio: percent of pixels to be above High threshold (suggested 0.99)
Smooth: Smoothing factor (suggested 0.9)
WindowSize: Size of window for adaptive gradient (suggested 7)
ThinFactor: Thinning factor (suggested 0)
DoHysteresis : If True turn on the hysteresis thresholding (suggested True)

ImageEnProc

Unit ImageENProc
TImageEnProc

DECLARATION

```
procedure EdgeDetect_Sobel;
```

DESCRIPTION

Performs an edge detect using Sobel filter. The result is a gray scale bitmap: high values (closed to 255) are edges. When Location is ieTBitmap, the output pixel format is i24RGB, otherwise it is ie8g. To make result black/white (with 1=edge) it is suggested to call ConvertToBWThreshold using parameter -2 (Maximum Entropy Algorithm).

DECLARATION

```
property FiltersInitialDir: string;
```

DESCRIPTION

FiltersInitialDir contains the initial directory in the previews dialog for the 'user filters' effect.

Declaration

```
procedure GammaCorrect(Gamma: double; Channel: TIEChannels);
```

Description

GammaCorrect performs a gamma correction.

Gamma is the gamma correction value. A value of 1.0 causes no gamma correction processing. Channel is the channel where to apply the gamma.

Declaration

```
procedure HistEqualize(LoThresh, HiThresh: TRGB);
```

Description

This method equalizes the colors histogram of the selected region in the range LoThresh and HiThresh.

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure HSLvar(oHue, oSat, oLum: integer);
```

DESCRIPTION

This method changes Hue, Saturation and Luminosity of the selected region.

Parameter	Description
oHue	Offset to add to the Hue channel, from -180 to +180.
oSat	Offset to add to the Saturation channel, from -100 to +100.
oLum	Offset to add to the Luminosity channel, from -100 to +100.

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure HSVvar(oHue, oSat, oVal: integer);
```

DESCRIPTION

This method changes Hue, Saturation and Value of the selected region.

Parameter	Description
oHue	Offset to add to the Hue channel, from -180 to +180.
oSat	Offset to add to the Saturation channel, from -100 to +100.
oVal	Offset to add to the Value channel, from -100 to +100.

Unit ImageENProc

TImageEnProc

Declaration

```
procedure IntensityRGBAll(r, g, b: integer);
```

Description

This method changes the RGB channels of the current image or of the selected region. r, g and b are the channel offsets, from -255 to +255.

Declaration

```
procedure Intensity(LoLimit, HiLimit, Change: integer; UseAverageRGB: Boolean; DoRed, DoGreen, DoBlue: Boolean);
```

Description

Changes the value of a pixel by an amount determined by the input Red, Green and Blue channel RGB values individually, or as an averaged value. The change is graduated starting from LoLimit, with a maximum change applied to the midpoint between LoLimit and HiLimit, then reduced again to HiLimit. Typically this method can be used to effectively progressively brighten or darken any range of colors as requested without affecting colors outside the range. RGB values in the range of 0 to 127 are considered as darker, and 128 to 255 as lighter.

LoLimit - an integer between 0 and 255 that indicates the start point of the range.

HiLimit - an integer between 0 and 255 that indicates the end point of the range.

Change - an integer between -255 and 255 that indicates the maximum or minimum value to be applied to the midpoint, and used to determine the slope of the increment/decrement used over the range.

If UseAverageRGB is true, it uses the average value of the input Red, Green and Blue channels to determine the amount to change all RGB channels in the pixel.

DoRed
DoGreen
DoBlue – If doBlue is true, it applies the change to this RGB channel.

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure Lens(cx, cy, Width, Height: integer;
Refraction: double);
```

DESCRIPTION

Lens applies the lens effect to the current image. This effect is available in image processing preview dialog.

Parameter	Description
cx	Horizontal lens position.
cy	Vertical lens position.
Width	Lens width.
Height	Lens height.
Refraction	Lens refraction (from 1).

ImageEnProc

DECLARATION

```
procedure MapGrayToColor(map: array of TRGB);
```

DESCRIPTION

MapGrayToColor maps each gray scale level to specified RGB color, using provided map.

Parameter	Description
map	Array of 256 elements of TRGB values.

DECLARATION

```
procedure Maximum(WindowSize: integer);
```

DESCRIPTION

The method Maximum sets each pixel in the output image to the maximum value of all the current image pixel values in the neighborhood of size WindowSize.

The maximum filter is typically applied to an image to remove negative outlier noise.

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure MedianFilter(WindowX: integer=5;
WindowY: integer=5; Brightness: integer=50;
Contrast: integer=50; Multiplier: integer=1;
Threshold: integer=50; MedianOp: TIEMedFilType =
mfMedianFilter);
```

DESCRIPTION

Carry out fast median filtering on images using windows from 3x3 to 19 x 19 maximum sizes. MedianFilter works only with true color images.
Filtering can be of three types:

1) Substitute median if central point differs from median by a threshold amount,
2) High pass sharpening
3) Edge extraction
4) Operation:

   Histogram in moving window at beginning of line is initiated. Histogram is updated after each move subtracting left column values from previous window, adding right column values. Median updated by counting up or down number of pixels less than or greater than old median.
5) For color images, the grayscale intensity of a pixel is computed as a weighted linear combination of RGB and counted in the histogram. The pixel with the nearest (L1 norm) color to the average of all pixels in the moving window except the central point is used in place of the median.

ImageEnProc

6) The computation of the median for a color image has a complexity of O(N^4), Whereas this technique is only O(N^2) or less and is as good when the window size is 19x19 when the median differs only by a small amount from the mean. For smaller windows performance degrades as the mean differs from the median, but it is still satisfactory down to a 5x5 window.

7) Adaptive thresholding is used to preserve sharp edges. A pixel is replaced by the median or its nearest average color equivalent if it lies outside the 1st and 3rd quartile of the intensity distribution. The user may modify the position of the quartiles interactively. Defaults are the first and third quartiles. For grayscale images, the quartiles and medians are used directly. The green element of a TRGB record is used as an intensity measurement. The user must provide means of detecting whether or not an image is color or grayscale.

Author:

I.Scollar, following Huang, Yang, Tang, unpublished report submitted under Defense Advanced Research Projects Agency contract

ImageEnProc

no. MDA 903-77-G-1, “A fast two dimensional median filtering algorithm”
T.S.Huang, G.J.Yang, G.Y.Tang, School of Electrical Engineering, Purdue University, West Lafayette, Indiana 47907, USA.

First publication: T.S.Huang, G.J.Yang, G.Y.Tang, Proceedings IEEE Conference Pattern Recognition and Image Processing, Chicago 1978, p. 128 ff.

I.Scollar, B.Weidner, T.S.Huang, Image enhancement using the median and the interquartile distance, Computer Vision, Graphics and Image Processing, 25 1984 236-251

Original Fortran IV, 512x512 grayscale images G.Y.Tang, 1976
Modifications:
Large images on DEC PDP11, I. Scollar Sept. 1978
Normalized sharpening, I.Scollar Sept. 1980
Adaptive quartile thresholding, I.Scollar Sept. 1980
Ported from DEC Fortran IV to Delphi 7 Pascal, I.Scollar March 11, 2003
Extension to color images, I. Scollar, March 13, 2003

ImageEnProc

Declaration

```
procedure MergeIEBitmap(DBitmap: TIEBitmap; pcf: integer);
```

Description

MergeIEBitmap merges the current image with a specified bitmap (like merge but works with TIEBitmap instead of TBitmap).

DBitmap is the bitmap to merge with the current image.
pcf is the percentage of current image (100=only current image, 0=only DBitmap).

Declaration

```
procedure Merge(DBitmap: TBitmap; pcf: integer);
```

Description

The Merge method merges the current image with a specified bitmap.

DBitmap is the bitmap to merge with current image.

Pcf is the percentage of current image (100=only current image, 0=only DBitmap).

ImageEnProc

Unit ImageENProc

TImageEnProc

Declaration

```
function MatchHSVRange(HueBegin, HueEnd,
SatBegin, SatEnd, ValBegin, ValEnd: integer;
ColorizeMatched: Boolean; MatchColor: TRGB;
ColorizeNonMatched: Boolean; NonMatchColor:
TRGB):integer;
```

Description

If ColorizeMatched is true then this method colorizes pixels that match the specified HSV (Hue, Saturation and Value) range to MatchColor color. If ColorizeNonMatched is true, non-matching pixels are set to NonMatchColor.

ImageEnProc

Parameter	Description
HueBegin	Starting Hue value. 0..359
HueEnd	Ending Hue value. 0..359
SatBegin	Starting Saturation value. 0..99
SatEnd	Ending Saturation value. 0..99
ValBegin	Starting Intensity value. 0..99
ValEnd	Ending Intensity value. 0..99
ColorizeMatched	If true next parameter specifies the matched color.
MatchColor	New pixel color when HSV conversion fit inside specified ranges.
ColorizeNonMatched	If true next parameter specifies the non-matched color.
NonMatchColor	New pixel color when HSV conversion does not fit inside specified ranges.

MATCHHSVRANGE RETURNS THE NUMBER OF MATCHING PIXELS.

ImageEnProc

Declaration

```
procedure Minimum(WindowSize: integer);
```

Description

The method Minimum sets each pixel in the output image as the minimum value of all the current image pixel values in the neighborhood of size WindowSize.

The minimum filter is typically applied to an image to remove positive outlier noise.

Declaration

```
procedure MotionBlur(angle: double; radius:
integer = 8; sigma: integer = 7);
```

Description

MotionBlur applies a motion blur effect.

Parameter	Description
angle	Specifies angle in degrees (0..360).
radius	Radius of Gaussian kernel in pixels (>0).
sigma	Standard deviation of Gaussian kernel in pixels (>0).

Declaration

```
procedure Negative;
```

Description

Negative method inverts all colors of the selected region.

Declaration

```
procedure Opening(WindowSize: integer);
```

Description

The method Opening performs an erosion (Minimum filter with WindowSize parameter) followed by a dilation (Maximum filter with WindowSize parameter).

The opening filter operation will reduce small positive oriented regions and positive noise regions.

Unit ImageENProc

TImageEnProc

Declaration

```
procedure Random(mean: double = 0.5; stdDev:
double = 0.123);
```

Description

Fills selected pixels with random values (Gaussian distribution). Random values are multiplied by 255 and truncated to the range 0..255.

Parameter	Description
mean	Mean value. 0.5 = gray (128).
stdDev	Standard deviation about mean.

Declaration

```
procedure RemoveRedEyes;
```

Description

RemoveRedEyes applies a simple algorithm which helps to remove red eyes. You can apply this function to the whole image, but it is better to select only the area which contains the eye, otherwise other parts of the image could be changed.

ImageEnProc

Declaration

```
procedure Sharpen(Intensity:integer = 10;
Neighbourhood: integer = 4);
```

Description

Sharpen performs sharpening filter.

Intensity specifies the intensity of the sharpening (1..100).

Neighbourhood specifies the window size.

Declaration

```
procedure Threshold(DownLimit, UpLimit, DownVal,
UpVal: TRGB);
```

Description

The Threshold method assigns the DownVal color to all colors smaller or equal to DownLimit, and UpVal to all colors greater than UpLimit.

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure Threshold2(LoThreshold, HiThreshold:
integer; Red, Green, Blue: Boolean);
```

DESCRIPTION

This is a variant of original Threshold algorithm.
LoThreshold and HiThreshold specify the low and high pixel value. Red, Green and Blue specify where to apply the threshold operation.

DECLARATION

```
procedure UnsharpMask(Radius: integer = 4;
Amount: double = 1.0; Threshold: double = 0.05);
```

DESCRIPTION

UnsharpMask subtracts the image to its Gaussian blurred version.

Parameter	Description
Radius	Radius of Gaussian blur matrix. (1..)
Amount	Difference between original and blur image. (0.0 to 5.0)
Threshold	Threshold of maximum luminosity value to apply the effect. (0.0 to 1.0)

ImageEnProc

Unit ImageENProc
TImageEnProc

DECLARATION

```
procedure WallisFilter(WinWidth: integer=2;
WinHeight: integer=2; Mean: integer=50; StDev:
integer=50; InGain: integer=50; Edge: integer=10;
Limit: integer=50; Thresholding: Boolean=false);
```

DESCRIPTION

WallisFilter works only with true color images.
Locally adaptive image enhancement based on:

Wallis, R. An approach to the space variant restoration and enhancement of images. Proceedings, Symposium on Current Mathematical Problems in Image Science, Monterey CA, 1976 10-12 reprinted in C.O. Wilde E. Barrett, eds. Image Science Mathematics, Western Periodicals, North Hollywood, CA, 1977. Summarized in Pratt:

Pratt, W.K. Digital Image Processing, 2nd. ed. John Wiley & Sons, New York, 1991, 248, 503

ImageEnProc

Unit ImageENProc
TImageEnProc

Programming for Delphi:

I. Scollar 2003

WALLIS - STATISTICAL DIFFERENCING USING WALLIS ALGORITHM

PURPOSE:

IMPLEMENT THE STATISTICAL DIFFERENCING FILTER OF WALLIS (SEE
W. PRATT, DIGITAL IMAGE PROCESSING) FOR LARGE PICTURES
AND LARGE WINDOWS GREATER THAN 20X20

EFFICIENT SAMPLING TECHNIQUE TO COMPUTE LOCAL MEANS AND
VARIANCES.

OPERATION:

THE WINDOW DIMENSIONS AND 5 PARAMETERS FOR THE WALLIS
FORMULA:
MEAN - DESIRED LOCAL MEAN OF OUTPUT
A COMMONLY USED VALUE FOR MANY APPLICATIONS
IS 128.ALLOWED RANGE 0-255.
S.D. DESIRED LOCAL STANDARD DEVIATION (CONTRAST)
OF OUTPUT PICTURE. ALLOWED RANGE 0-100.
COMMONLY USED RANGE 50-75.
GAIN CONTROLS RATIO OF MEASURED TO DESIRED
LOCAL VARIANCE. ALLOWED RANGE 0-INFINITY
EFFECT:
0 NO VARIANCE EQUALIZATION
THIS IS THE SAME AS LEE'S
ALGORITHM (IEEE PROC. PRIP 1978,P.56)

INF MAXIMUM VARIANCE EQUALIZATION
THIS IS EQUIVALENT TO WATKIN'S
ALGORITHM.
COMMONLY USED RANGE 4-25.

EDGE FACT. CONTROLS AMOUNT OF MEAN EQUALIZATION
ALLOWED RANGE 0-1.
EFFECT:
100 FULL MEAN EQUALIZATION
0 MEASURED MEAN RESTORED

AUTHORS:

B. WEIDNER, RLMB, 1979
BASED ON IDEAS TAKEN FROM MVEQN/F BY I. SCOLLAR 1979
BASED ON IDEAS TAKEN FROM ANAY14 BY G. TANG, 1977

Unit ImageENProc

TImageEnProc

Declaration

```
procedure Wave(amplitude, wavelength, phase: integer; reflective: Boolean);
```

Description

Wave applies a wave effect to the current image. This effect is available in the image processing preview dialog.

Parameter	Description
amplitude	Amplitude (height) of the wave (from 0).
wavelength	Length of the wave (from 0).
phase	phase of the wave in degrees (0..359).
reflective	If true makes a special effect.

Declaration

```
procedure WhiteBalance_coef(Red, Green, Blue: double);
```

Description

This method applies specified coefficients to each pixel of the image. Coefficients (Red, Green, and Blue) are values in the range 0 to 1.

ImageEnProc

Declaration

```
procedure WhiteBalance_WhiteAt(WhiteX, WhiteY: integer);
```

Description

Adjust image colors adjusting the white range. WhiteX and WhiteY are the coordinates of a white pixel.

Clipboard

Declaration

```
function CopyToClipboard(UseInternalFormat: Boolean): Boolean;
```

Description

This method copies whole image to the clipboard.
If UseInternalFormat is true, two bitmap are saved: one in standard DIB format and one in internal format which preserves pixelformat and alpha channel. CopyToClipboard returns true on successful.

Unit ImageENProc

TImageEnProc

DECLARATION

```
function IsClipboardAvailable: Boolean;
```

DESCRIPTION

This method returns true if clipboard contains data valid for TImageEnProc.

See also
PasteFromClipboard
PointPasteFromClip
SelPasteFromClipStretch
SelPasteFromClip

Use of this function eliminates often encountered Clipboard not available messages when using vcl clipboard methods.

ImageEnProc

Declaration

```
procedure PasteFromClipboard;
```

Description

This method makes the clipboard image as the current image. All uncompressed DBI formats are handled (1, 4, 8, 15, 16, 24 and 32 bits). If the clipboard image is 4, 8, 15, 16, 24 or 32 bits, it is converted to 24 bit. Otherwise (1 bit) it remains 1 bit.

Declaration

```
procedure PointPasteFromClip(x1, y1: integer;
MergeAlpha: Boolean=true);
```

Description

This method pastes the clipboard image at x1, y1 position (component coordinates). All uncompressed DBI formats are handled (1, 4, 8, 15, 16, 24 and 32 bits). If clipboard image is 4, 8, 15, 16, 24 or 32 bits, it is converted to 24 bit, otherwise (1 bit) it remains 1 bit.

MergeAlpha is valid only transferring image in internal format. It merges the pasting bitmap alphachannel with the background bitmap.

Unit ImageENProc

TImageEnProc

Declaration

```
function SelCopyToClip(UseInternalFormat:
Boolean): Boolean;
```

Description

This method copies the selected region to the clipboard.
If UseInternalFormat is true, two bitmap are saved: one in standard DIB format and one in internal format which preserves pixelformat and alpha channel. SelCopyToClip returns true on successful.

Declaration

```
function SelCutToClip(UseInternalFormat:
Boolean): Boolean;
```

Description

This method copies and cuts the selected region to the clipboard. If UseInternalFormat is true, two bitmap are saved: one in standard DIB format and one in internal format which preserves pixelformat and alpha channel. SelCutToClip returns true on successful.

Declaration

```
procedure SelPasteFromClipStretch(MergeAlpha:
Boolean=true);
```

Description

This method pastes the clipboard image to the selected region, stretching the image to fit the selection. All uncompressed DBI formats are handled (1, 4, 8, 15, 16, 24 and 32 bits). If clipboard image is 4, 8, 15, 16, 24 or 32 bits, it is converted to 24 bit, otherwise (1 bit) it remains 1 bit.

MergeAlpha is valid only transferring image in internal format. It merges the pasting bitmap alphachannel with the background bitmap.

Declaration

```
procedure SelPasteFromClip(MergeAlpha:
Boolean=true);
```

Description

This method pastes the contents of the Clipboard inside the selected area. The image will be stretched to the selection size. MergeAlpha is valid only transferring image in internal format. It merges the pasting bitmap alphachannel with the background bitmap.

Steganography

DECLARATION

```
procedure ClearHiddenText;
```

DESCRIPTION

This method removes the text added using WriteHiddenText or WriteHiddenData.

DECLARATION

```
function GetHiddenDataSpace: integer;
```

DESCRIPTION

GetHiddenDataSpace returns the available hidden space inside the current image (in bytes).

ImageEnProc

Unit ImageENProc
TImageEnProc

Declaration

```
function ReadHiddenData(data: PAnsiChar; maxlen: integer):integer;
```

Description

ReadHiddenText or ReadHiddenData returns hidden text or hidden raw data written with WriteHiddenText or WriteHiddenData.

Hidden information is stored inside the image (uses a pixel color modulation) and is independent of the image file format. The hidden text will be lost if you save an image with Jpeg or subsample colors. Use ReadHiddenText to read simple string and ReadHiddenData to read buffer of raw data. If you set data to nil and maxlen to 0 ReadHiddenData returns the length of data to read.

ImageEnProc

Unit ImageENProc
TImageEnProc

DECLARATION

```
function ReadHiddenText: AnsiString;
```

DESCRIPTION

ReadHiddenText or ReadHiddenData returns hidden text or hidden raw data written with WriteHiddenText or WriteHiddenData.

Hidden information is stored inside the image (uses a pixel color modulation) and is independent of the image file format. The hidden text will be lost if you save an image with Jpeg or subsample colors. Use ReadHiddenText to read simple string and ReadHiddenData to read buffer of raw data. If you set data to nil and maxlen to 0 ReadHiddenData returns the length of data to read.

ImageEnProc

Unit ImageENProc
TImageEnProc

DECLARATION

```
function WriteHiddenData(data: PAnsiChar; count: integer):integer;
```

DESCRIPTION

WriteHiddenText or WriteHiddenData writes hidden text or hidden raw data in a true color image, respectively. Hidden information is stored inside the image (uses a pixel color modulation) and is independent of the image file format.

The hidden text will be lost if you save the image with Jpeg or if you subsample the colors. Use WriteHiddenText to write a simple string and WriteHiddenData to write a block of count bytes (you can write to a hidden image or sound).

ImageEnProc

Unit ImageENProc

TImageEnProc

Declaration

```
function WriteHiddenText(text:
AnsiString):integer;
function WriteHiddenData(data: PAnsiChar; count:
integer):integer;
```

Description

WriteHiddenText or WriteHiddenData writes hidden text or hidden raw data in a true color image, respectively. Hidden information is stored inside the image (uses a pixel color modulation) and is independent of the image file format.

The hidden text will be lost if you save the image with Jpeg or if you subsample the colors. Use WriteHiddenText to write a simple string and WriteHiddenData to write a block of count bytes (you can write to a hidden image or sound).

ImageEnProc

Encrypting

Declaration

```
procedure Decrypt(Passkey: AnsiString);
procedure Decrypt(Passkey: array of byte);
```

Description

This function decrypts current image (layer), crypted using Encrypt. The algorithm used is TEA Tiny Encryption Algorithm with a key of 128 bits. Using PassKey as a string you should specify a string password, which will be hashed to a 128 bits key. Using PassKey as array of bytes you should specify a binary key of 16 bytes.

Unit ImageENProc

TImageEnProc

Declaration

```
procedure Encrypt(Passkey: AnsiString);
procedure Encrypt(Passkey: array of byte);
```

Description

This function encrypts current image (layer).

The image should be saved using lossless formats and in full rgb color spaces (no palette).

To decrypt an image use Decrypt. The algorithm used is TEA Tiny Encryption Algorithm with a key of 128 bits. Using PassKey as a string you should specify a string password, which will be hashed to a 128 bits key.

Using PassKey as array of bytes you should specify a binary key of 16 bytes. There is no way to know when an image is crypted, unless you insert special tags (like EXIF or IPTC) manually.

ImageEnProc

Geometric

Declaration

```
function AutoCrop(Tolerance: integer; Background: TRGB; DoCrop: Boolean):TRect;
function AutoCrop(Tolerance: integer; Background: TColor; DoCrop: Boolean):TRect;
```

Description

AutoCrop removes the border with color like Background from the image borders. The Tolerance specifies how much background color must be equal to actual pixel. When optional DoCrop parameter is False then AutoCrop doesn't crop the image, but just return the suggested area to crop.

AutoCrop returns the area to crop or cropped.

ImageEnProc

Declaration

```
function AutoCrop2(BorderRate: double=6; DoCrop: Boolean): TRect;
```

Description

AutoCrop2 automatically removes the border using a density analysis algorithm. BorderRate is the difference between border and text. Must be >0. This method works only when paper is white and text is black. When optional DoCrop parameter is False then AutoCrop2 doesn't crop the image, but just return the suggested area to crop. AutoCrop2 returns the area to crop or cropped.

Declaration

```
procedure CropSel(TransparencyOnly: Boolean=false);
```

Description

CropSel cuts the selected region and copies as current image, without use clipboard. When TransparencyOnly is true only the alpha channel is cut.

ImageEnProc

Unit ImageENProc

TImageEnProc

Declaration

```
procedure Crop(x1, y1, x2, y2:integer);
procedure Crop(Rect: TRect);
```

Description

Crop gets as current image the specified rectangle, cutting off the remaining image.

Declaration

```
procedure Flip(dir: TFlipDir);
```

Description

This method flips (mirrors) the current image across the horizontal or vertical axis. Flipping is not the same as rotation. Dir is the flip direction.

Unit ImageENProc

TImageEnProc

Declaration

```
procedure ImageResize(newWidth, newHeight:
integer; HorizAlign: TIEHAlign=iehLeft;
VertAlign: TIEVAlign=ievTop; FillAlpha:
integer=255);
```

Description

This method resizes the current image to newWidth and newHeight. The content of the image doesn't change (no stretch). HorizAlign specifies the horizontal alignment of the old image. VertAlign specifies the vertical alignment of the old image. FillAlpha specifies the transparency to use on added areas.

Declaration

```
procedure MakeTile(columns, rows: integer);
```

Description

MakeTile replicates current image for the specified columns and rows numbers. The final image will be resized by columns*ImageWidth and rows*ImageHeight. This is useful to create tiled images.

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure PerspectiveDraw(Source: TIEBitmap; x0,
y0, x1, y1, x2, y2, x3, y3:integer; alphaMin:
integer=-1; alphaMax: integer=-1; mergeAlpha:
Boolean);
```

DESCRIPTION

Draws source bitmap over current layer, stretching bitmap inside four points.

Parameter	Description
x0, y0	left/top point
x1, y1	right/top point
x2, y2	right/bottom point
x4, y3	left/bottom point
alphaMin, alphaMax	If alphaMin>-1 and alphaMax>-1 then alpha (transparency) will be set in the specified range. This is useful to compose “cover-flow” like reflection. alphaMin (0..255) specifies minimum transparency. alphaMax (0..255) specifies maximum transparency.
mergeAlpha	If image alpha channel is merged with the background image.

It is possible to invert the image making, for example, (x0,y0) > (x1,y1).

ImageEnProc

Unit ImageENProc

TImageEnProc

Declaration

```
function ProjectDraw(Source: TIEBitmap;
centerDestX: integer; centerDestY: integer;
destWidth: integer; destHeight: integer; depth:
double; translateX: integer; translateY: integer;
rotateX: double; rotateY: double;
specularAlphaMin: integer; specularAlphaMax:
integer; mergeAlpha: Boolean = false):
TIEQuadCoords;
function ProjectDraw(Source: TIEBitmap;
centerDestX: integer; centerDestY: integer;
scale: double; depth: double; translateX:
integer; translateY: integer; rotateX: double;
rotateY: double; specularAlphaMin: integer;
specularAlphaMax: integer; mergeAlpha: Boolean =
false): TIEQuadCoords;
```

Description

Draws source bitmap over current layer, performing translations, rotations and perspective transform.

Parameter	Description
Source	Source bitmap to draw.
centerDestX	Horizontal position of center of destination position.
centerDestY	Vertical position of center of destination position.
destWidth	Destination bounding box width, not including specular bitmap. The resulting bitmap will be adapted (but not stretched) to this size.

Parameter	Description
destHeight	Destination bounding box height, not including specular bitmap. The resulting bitmap will be adapted (but not stretched) to this size.
scale	Replaces destWidth and destHeight in the second overload. Specifies a multiplier of the original size (1=same size, 0.5=half size, etc..).
depth	Destination depth of field.
translateX	Horizontal offset in 3D space (before perspective projection)
translateY	Vertical offset in 3D space (before perspective projection)
rotateX	X axis rotations in 3D space (before perspective projection). Angles are in degrees.
rotateY	Y axis rotations in 3D space (before perspective projection). Angles are in degrees.
specularAlphaMin	If >-1 a reflection bitmap will be drawn, using specified alpha range.
specularAlphaMax	If >-1 a reflection bitmap will be drawn, using specified alpha range.
mergeAlpha	If image alpha channel is merged with the background image.

Unit ImageENProc

TImageEnProc

Declaration

```
procedure RadialStretch(ARed, BRed, CRed, DRed,
AGreen, BGreen, CGreen, DGreen, ABlue, BBlue,
CBlue, DBlue: double);
```

Description

This function performs a radial stretch for each color component (R, G, and B). This allows manual correction of Barrel Distortion and Pincushion distortion -lens distortion, underwater distortion. A, B, C and D (followed by channel name) are the coefficients

Look at the demo for more details.

Declaration

```
procedure Reflection(minAlpha: integer = 0;
maxAlpha: integer=200; percentage: integer=100);
```

Description

Reflection extends image vertically to simulate a reflection (like reflection of “coverflow” presentations).

Parameter	Description
minAlpha	Minimum alpha value (0=fully transparent, 255=fully opaque).
maxAlpha	Maximum alpha value (0=fully transparent, 255=fully opaque).
percentage	Effect percentage (0=no effect, 100=maximum effect).

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure Resample(NewWidth, NewHeight: integer;
FilterType: TResampleFilter);
```

DESCRIPTION

This method resizes the current image. The content of the image changes (stretched to new size).

Parameter	Description
NewWidth	New image width in pixels. If NewWidth is -1 then it is calculated automatically, respecting the proportions.
NewHeight	New image height in pixels. If NewHeight is -1 then it is calculated automatically, respecting the proportions.
FilterType	Resampling interpolation algorithm.

See also

ResampleTo.

Unit ImageENProc
TImageEnProc

DECLARATION

```
procedure ResampleTo(Target: TIEBitmap;
TargetWidth, TargetHeight: integer; FilterType:
TResampleFilter);
```

DESCRIPTION

ResampleTo copies a resampled instance of the current image to a Target TIEBitmap. The content of the destination image changes (stretched to new size).

Parameter	Description
Target	The target (destination) image.
TargetWidth	Width in pixels of the target size. If TargetWidth is -1 then it is calculated automatically, respecting the proportions.
TargetHeight	Height in pixels of the target size. If TargetHeight is -1 then it is calculated automatically, respecting the proportions.
FilterType	Resampling interpolation algorithm.

Resampling black/white (1bit) images with FilterType is not rfNone, Resample converts the image to 24bit.

See also
Resample.

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure RotateAndCrop(Angle: double; antialias:
Boolean = true; AntialiasMode: TIEAntialiasMode =
ierFast);
```

DESCRIPTION

This function rotates the image by Angle (in degrees) and crops the borders of the original bitmap. It is useful to deskew an image knowing the rotation angle. Angle must be in the range -90 and +90 degrees. Other parameters are like Rotate method. If you have this image you can know the rotation needed to deskew it executing: D:=ImageEnView1.Proc.SkewDetection;

Now you can rotate the image executing ImageEnView1.Proc.Rotate(D), but you get:

The gray border (which is part of the original bitmap) is rotated with the image.

If you need to remove that border you have to execute ImageEnView1.Proc.RotateAndCrop(D), so you will have:

ImageEnProc

Unit ImageENProc

TImageEnProc

Declaration

```
procedure Rotate(fangle: double; antialias:
Boolean = false; AntialiasMode: TIEAntialiasMode
= ierFast; BackgroundColor: TColor = -1);
```

Description

The Rotate method rotates the current image of fangle angle (negative or positive degrees). If antialias is True, an anti-alias algorithm is used. Applications can select the anti-alias algorithm using AntialiasMode parameter, values of this parameter can be: ierFast : fast but with low quality ierBilinear : bilinear, high quality ierBicubic : bicubic, high quality (maximum quality) BackgroundColor specifies an alternative background color. If it is -1, Rotate uses the default background.

Declaration

```
procedure RoundImage(RoundWidth, RoundHeight:
integer);
```

Description

Makes corners rounded. RoundWidth and RoundHeight specify the round size.

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure ShiftChannel(offsetX, offsetY: integer;
channel: integer; fillValue: integer);
```

DESCRIPTION

ShiftChannel moves the specified channel by an offset filling blank position with a color.

Parameter	Description
offsetX	Specifies the horizontal offset in pixels. This can have negative values.
offsetY	Specifies the vertical offset in pixels. This can have negative values.
channel	The channel to shift. 0=Blue, 1=Green and 2=Red.
fillValue	The filling color (this is the channel intensity, because ShiftChannel works on a channel at the time).

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
function SkewDetectionFine(StartingAngle: double;
resolution: double; range: integer; maxQuality:
Boolean):double;
```

DESCRIPTION

SkewDetectionFine estimates the orientation angle (in degrees) of the lines of text. Apply this method only to images that contains printed text. SkewDetectionFine doesn't use a Hough transform (like SkewDetection), but progressive rotations to find the best orientation.

StartingAngle specifies the starting angle, if you know it (0 otherwise). Resolution is the angle increment (0.1 is good).
Range specifies the angle range (example 10, try from -5 to +5).
MaxQuality : if False, creates a thumbnail of the image and works on it.

ImageEnProc

Unit ImageENProc

TImageEnProc

Declaration

```
function SkewDetection(ResampleWidth: integer;
AngleRange: integer; Precision: double;
EdgeDetect: Boolean):double;
```

Description

SkewDetection estimates the orientation angle (in degrees) of the lines of text. Apply this method only to images that contains printed text. If there is a selected area, SkewDetection works only on this one. SkewDetection can be very slow on large images, so applications can resample analyzed image using ResampleWidth parameter to enhance performance. No resampling is performed when ResampleWidth is zero. AngleRange specifies the working range in degrees. Low values enhance performance and accuracy. For example specifying 30 SkewDetection will scan the image searching for pixels inside a range of -15 to 15 degrees. AngleRange can be from 1 to 180. Precision specifies the angle precision in degrees. It must be >0 and <=1 with only one decimal digit. Typical value is 0.1.

EdgeDetect specifies if to apply and edge detect algorithm before detect the orientation.

ImageEnProc

Automatic Image Enhancement

DECLARATION

```
procedure AdjustGainOffset;
```

DESCRIPTION

Adjust image luminosity calculating the min and max pixels values and stretching colors to the maximum allowed value.

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure AutoImageEnhance1(SubsampledSize:
integer; Slope: integer; Cut: integer;
Neightbour: integer);
```

DESCRIPTION

This is the first of a group of functions already implemented or that will be implemented in future which helps to perform an automatic image adjustment. AutoImageEnhance1 performs complex operations on the image to simulate how humans see the world to make the image to have better contrast and colors.

Parameters

SumsamplesSize: to make it fast operations are performed on a subsampled image. This parameter specifies the image width. (Default is 60).

Slope: slope specifies the core function slope and it is measured in degrees. This value contributes to change contrast. Allowed values 0..90°, default 20. Cut: cut specifies the code function range. This value contributes to change contrast. Allowed values are 0..100, default 25. Neightbour: when the subsampled image is used to change the full size image. Neighbour specifies the window size of the conversion. High values slow down process but produce better results. Default is 2.

The default parameters were good for our set of test images.

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure AutoImageEnhance2(ScaleCount: integer;
ScaleCurve: integer; Variance: double; ScaleHigh:
integer; Luminance: Boolean);
```

DESCRIPTION

AutoImageEnhance2 performs complex operations on the image to simulate how humans see the world to make the image to have better contrast and colors.

Parameter	Description
ScaleCount	The algorithm is applied to different scales of the image. This parameter specifies the number of scales. Default 3.
ScaleCurve	Specifies how build the size of each scale. Allowed values 0..2. Default 2.
Variance	The output variance. Default 1.8.
ScaleHigh	Allowed values from 16 to 250. Default 200.
Lumincance	If true the algorithm is applied only to luminance (luminosity) channel, so the algorithm doesn't touch colors.

The default parameters were good for our set of test images.

ImageEnProc

Declaration

```
procedure AutoImageEnhance3(Gamma: double;
Saturation: integer);
```

Description

AutoImageEnhance3 enhances luminosity applying a local adaptation tone mapping algorithm (thanks to Prof. Irwin Scollar). Gamma adjusts resulting luminosity. Allowed values > 0. Saturation adjusts color saturation and allows values from -100 to 100.

Description

```
procedure AutoSharp(Intensity: integer; rate:
double);
```

Description

AutoSharp enhances image sharpness, increasing the sharping of detected objects contours. Intensity (0..100) controls the intensity of the effect (default 68). Rate (0..1) controls the borders detection rate (default 0.035).

Declaration

```
procedure HistAutoEqualize;
```

Description

This method equalizes the colors histogram of the selected region.

Declaration

```
procedure WhiteBalance_AutoWhite;
```

Description

This procedure adjusts image colors adjusting the white range. This method is like WhiteBalance_WhiteAt, but tries to find the white automatically.

Declaration

```
procedure WhiteBalance_GrayWorld;
```

Description

Adjust image colors applying the white balance algorithm named Gray World.

Undo

Declaration

```
property AutoUndo: Boolean;
```

Description

If AutoUndo is True, all image processing and load operations automatically call the SaveUndo method.

Unit ImageENProc

TImageEnProc

Declaration

```
property CanUndo: Boolean;
```

Description

CanUndo is true when the undo-buffer contains an image. ClearAllUndo or ClearUndo if there is only one saved image sets CanUndo to false.

Declaration

```
procedure ClearAllUndo;
```

Description

ClearAllUndo removes all images in the Undo stack.

Declaration

```
procedure ClearUndoAt(Position: integer);
```

Description

ClearUndoAt removes the image at Position in the undo-stack. Position: 0=last saved undo, 1=before last saved undo, 2... up to UndoCount-1.

ImageEnProc

Declaration

```
procedure ClearUndo;
```

Description

The ClearUndo method empties the last undo buffer and frees memory allocated for it.

Declaration

```
procedure SaveUndoCaptioned(const Caption:
string; Source: TIEUndoSource = ieuImage);
```

Description

SaveUndoCaptioned is like SaveUndo, but allows specifying a description associated to the saved image. Source specifies if the image (ieuImage), the current selection (ieuSelection) or vectorial objects (ieuObject) or layer is to be saved.

Declaration

```
procedure SaveUndo(Source:
TIEUndoSource=ieuImage);
```

Description

This method saves the current image in the undo-buffer list. Source specifies if the image (ieuImage), the selected region (ieuSelection) or vectorial objects (ieuObject) is saved or layers info (ieuLayer)

Unit ImageENProc

TImageEnProc

Declaration

```
procedure UndoAt(Position: integer; AutoRedo:
Boolean=false);
```

Description

UndoAt restores the image at Position in the Undo stack. Position: 0=last saved undo, 1=before last saved undo, 2... up to UndoCount-1. When AutoRedo is true the current state (before redo) is saved in redo-stack.

Declaration

```
property UndoCaptions[index: integer]:string;
```

Description

Applications can associate a string with saved images in the undo-stack. All ImageEn image processing functions save a little description of the last task performed. The UndoCaptions property is useful to build a list of Undo/Redo with description of the all functions executed. Index: 0=last saved undo, 1=before last saved undo, 2... up to UndoCount-1.

Declaration

```
property UndoCount: integer;
```

Description

UndoCount returns how many images there are in the undo-stack.

Declaration

```
property UndoLimit: integer;
```

Description

UndoLimit specifies how many images can be saved using the SaveUndo method. Default is 1.

When you call SaveUndo ImageEn pushes the current image in an image stack. Calling Undo ImageEn restore last saved image. Calling ClearUndo ImageEn removes last saved image.

Declaration

```
property UndoLocation: TIELocation;
```

Description

UndoLocation specifies where ImageEn saves 'Undo' images. Default is ieFile.

Declaration

```
function UndoPeekAt(Position: integer):
TIEUndoSource;
```

Description

UndoPeekAt returns the type of undo located at Position. It can be an image or a selection.

ImageEnProc

DECLARATION

```
procedure UndoRect(x1, y1, x2, y2:integer);
```

DESCRIPTION

This is like Undo but restores only the specified rectangle.
This is useful when you are performing interactive user drawing like in icon editor example or 'brush' example.

DECLARATION

```
procedure Undo(AutoRedo: Boolean=false);
```

DESCRIPTION

This method makes the last Undo buffer as the current image.
When AutoRedo is true the current state (before redo) is saved in redo stack.

Redo

DECLARATION

```
property CanRedo: Boolean;
```

DESCRIPTION

CanRedo is true when the Redo buffer contains an image.

Declaration

```
procedure ClearAllRedo;
```

Description

ClearAllRedo removes all images from the Redo stack.

Declaration

```
procedure ClearRedo;
```

Description

ClearRedo clears only the last undo entry.

Declaration

```
procedure RedoAt(Position: integer);
```

Description

RedoAt restores the image at Position in the redo-stack. Position: 0=last saved redo, 1=before last saved redo, 2... up to RedoCount-1.

Unit ImageENProc

TImageEnProc

DECLARATION

```
property RedoCaptions[index: integer]:string;
```

DESCRIPTION

Applications can associate a string to a saved image in the redo-stack. All ImageEn image processing functions save a little description. (RedoCaptions) of the last task performed. This property is useful to build a list of Undo/Redo with description of the all functions executed. Index: 0=last saved redo, 1=before last saved redo, 2... up to RedoCount-1.

DECLARATION

```
property RedoCount: integer;
```

DESCRIPTION

RedoCount returns how many images there are in the redo-stack.

ImageEnProc

Declaration

```
function RedoPeekAt(Position: integer): TIEUndoSource;
```

Description

RedoPeekAt returns the type of redo located at Position. It can be an image or a selection.

Declaration

```
procedure Redo;
```

Description

Redo makes last redo buffer as current image.

Declaration

```
procedure SaveRedoCaptioned(const Caption: string; Source: TIEUndoSource=ieuImage);
```

Description

SaveRedoCaptioned is like SaveRedo, but allows specification of a description associated to the saved image.

Source specifies if the image (ieuImage), the current selection (ieuSelection) or vectorial objects (ieuObject) or layer is to be saved.

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure SaveRedo(Source:
TIEUndoSource=ieuImage);
```

DESCRIPTION

SaveRedo saves the current image in the redo-buffer list. Source specifies if the image (ieuImage), the current selection (ieuSelection) or vectorial objects (ieuObject) or layer is to be saved.

ImageEnProc

Transitions

Declaration

```
procedure PrepareTransitionBitmaps(StartBitmap,
EndBitmap: TBitmap; Effect: TIETransitionType;
iWidth: Integer = -1; iHeight: Integer = -1;
BackgroundColor: TColor = -1; ResamplingFilter:
TResampleFilter);
```

Description

Use with CreateTransitionBitmap to create a series of frames that transition from StartBitmap to EndBitmap.

Parameter	Description
StartBitmap	The image that we transition from
EndBitmap	The image that we transition to
Effect	The desired transition effect
iWidth, iHeight	The size to create the transition bitmaps. If either of these are -1 then the size will be the larger of the two images in each dimension. Aspect Ratios will be maintained and any non-image area will be filled with BackgroundColor.
BackgroundColor	The color that will be used for blank frames or non-image area (if -1 then Background is used)

Parameter	Description
ResamplingFilter	The algorithm that is used to improve quality when resizing images

Note: Use PrepareTransitionBitmapsEx if you need to create frames for an iettPanZoom transition

See also
PrepareTransitionBitmapsEx
CreateTransitionBitmap

ImageEnProc

Example

```
procedure TransitionFrameCreationExample;
var
OldBitmap, NewBitmap, TransBitmap: TBitmap;
I: Integer;
TransLevel: Single;
begin
   OldBitmap := TBitmap.Create;
   NewBitmap := TBitmap.Create;
   TransBitmap := TBitmap.Create;
   try
      OldBitmap.LoadFromFile('C:\OldImage.bmp');
      NewBitmap.LoadFromFile('C:\NewImage.bmp');
      // Call PrepareTransitionBitmaps once
      ImageEnProc.PrepareTransitionBitmaps(
         OldBitmap, NewBitmap,
         iettCrossDissolve);
      for i := 1 to 9 do
      begin
         // Transition levels from 10% to 90%
         TransLevel := i * 10;
         // Call CreateTransitionBitmap for each
         // required frame
         ImageEnProc.CreateTransitionBitmap(
            TransLevel, TransBitmap);
         TransBitmap.SaveToFile('C:\TransImage' +
            IntToStr(I) + '.bmp');
      end;
      finally
         OldBitmap.Free;
         NewBitmap.Free;
         TransBitmap.Free;
      end;
end;
```

ImageEnProc

Unit ImageENProc

TImageEnProc

Declaration

```
procedure PrepareTransitionBitmapsEx(StartBitmap,
EndBitmap: TBitmap; Effect: TIETransitionType;
StartRect, EndRect: TRect;
RectMaintainAspectRatio: boolean = True;
iWidth: Integer = -1; iHeight: Integer = -1;
bStretchSmall: Boolean = False;
BackgroundColor: TColor = -1; ResamplingFilter :
TResampleFilter;
Timing: TIETransitionTiming = iettLinear);
```

Description

This is an extended version of PrepareTransitionBitmaps that includes more parameters and is primarily used when you need to create a series of frames that show a Pan Zoom from StartRect to EndRect for image StartBitmap.

Parameter	Description
StartBitmap	The image that we transition from
EndBitmap	The image that we transition to (NIL for a iettPanZoom transition)
Effect	The desired transition effect
StartRect	When using an iettPanZoom effect this is the portion of the image that is shown at the start
EndRect	When using an iettPanZoom effect this is the portion of the image that is shown at the end

Parameter	Description
RectMaintainAspectRatio	ImageEn will ensure that the starting and ending rects are automatically adjusted to ensure the resultant image has the correct aspect ratio (iettPanZoom only)
iWidth, iHeight	The size to create the transition bitmaps. If either of these are -1 then the size will be the larger of the two images in each dimension. Aspect Ratios will be maintained and any non-image area will be filled with BackgroundColor.
bStretchSmall	If the images are smaller than the transition bitmap size (iWidth x iHeight) should they be stretched to fit (which can lead to distortion).
BackgroundColor	The color that will be used for blank frames or non-image area (if -1 then Background is used)
ResamplingFilter	The algorithm that is used to improve quality when resizing images
Timing	The rate<L> at which the transition progresses

To create Pan Zoom transitions for an image call it as follows:

```
PrepareTransitionBitmapsEx(MyBitmap, nil,
   iettPanZoom, StartingRect, EndingRect);
CreateTransitionBitmap(TransitionLevel,
   MyPanZoomBitmap);
```

See also

PrepareTransitionBitmaps

CreateTransitionBitmap

Example

```
procedure PanZoomFrameCreationExample(StartingRect, EndingRect : TRect);
var
MyBitmap, PanZoomBitmap: TBitmap;
I: Integer;
TransLevel: Single;
begin
  MyBitmap := TBitmap.Create;
  PanZoomBitmap := TBitmap.Create;
  try
    MyBitmap.LoadFromFile('C:\MyImage.bmp');
    // Call PrepareTransitionBitmaps once
    ImageEnProc.PrepareTransitionBitmapsEx(
      MyBitmap, MyBitmap, iettPanZoom,
      StartingRect, EndingRect);
    for i := 0 to 10 do
    begin
      // Pan Zoom Transitions from StartingRect
      // (0%) to EndingRect (100%)
      TransLevel := i * 10;
     // Call CreateTransitionBitmap for each
     // required frame
     ImageEnProc.CreateTransitionBitmap(
       TransLevel, PanZoomBitmap);
     PanZoomBitmap.SaveToFile('C:\PanZoomImage' +
       IntToStr(I) + '.bmp');
    end;
    finally
      MyBitmap.Free;
      PanZoomBitmap.Free;
    end;
end;
```

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
procedure CreateTransitionBitmap(
TransitionProgress: Single; DestBitmap: TBitmap);
```

DESCRIPTION

Use with PrepareTransitionBitmaps or PrepareTransitionBitmapsEx to create a series of frames that transition from one bitmap to another.

Parameter	Description
TransitionProgress	The percentage that it is has progressed from the start image to the end image (ranging from 0.0 to 100.0)
DestBitmap	Will be filled with the created transition frame. It must be created before calling and will be automatically sized and set to 24 bit

See also

PrepareTransitionBitmaps

PrepareTransitionBitmapsEx

Example

```
procedure TransitionFrameCreationExample;
var
OldBitmap, NewBitmap, TransBitmap: TBitmap;
I: Integer;
TransLevel: Single;
begin
   OldBitmap := TBitmap.Create;
   NewBitmap := TBitmap.Create;
   TransBitmap := TBitmap.Create;
   try
   OldBitmap.LoadFromFile('C:\OldImage.bmp');
   NewBitmap.LoadFromFile('C:\NewImage.bmp');
   // Call PrepareTransitionBitmaps once
   ImageEnProc.PrepareTransitionBitmaps(OldBitmap,
      NewBitmap, iettCrossDissolve);
   for i := 1 to 9 do
   begin
      // Transition levels from 10% to 90%
      TransLevel := i * 10;
      // Call CreateTransitionBitmap for each
      // required frame
      ImageEnProc.CreateTransitionBitmap(TransLevel,
         TransBitmap);
      TransBitmap.SaveToFile('C:\TransImage' +
         IntToStr(I) + '.bmp');
   end;
   finally
      OldBitmap.Free;
      NewBitmap.Free;
      TransBitmap.Free;
   end;
end;
```

ImageEnProc

Unit ImageENProc

TImageEnProc

DECLARATION

```
TIETransitionType = (iettNone,
iettCrossDissolve, // Cross Fade
iettFadeOut, // Fade Out
iettFadeIn, // Fade In
iettFadeOutIn, // Fade Out then In
iettLeftRight1, // Wipe Left to Right
iettLeftRight2, // Wipe Left to Right 2
iettRightLeft1, // Wipe Right to Left
iettRightLeft2, // Wipe Right to Left 2
iettUpDown1, // Wipe Top to Bottom
iettUpDown2, // Wipe Top to Bottom 2
iettDownUp1, // Wipe Bottom to Top
iettDownUp2, // Wipe Bottom to Top 2
iettFromUpLeft, // Slide from Top Left
iettFromUpRight, // Slide from Top Right
iettFromBottomLeft, // Slide from Bottom Left
iettFromBottomRight, // Slide from Bottom Right
iettMoveLeftRight1, // Push Left to Right
iettMoveLeftRight2, // Slide Out Left to Right
iettMoveRightLeft1, // Push Right to Left
iettMoveRightLeft2, // Slide Out Right to Left
iettMoveUpDown1, // Push Top to Bottom
iettMoveUpDown2, // Slide Out Top to Bottom
iettMoveDownUp1, // Push Bottom to Top
iettMoveDownUp2, // Slide Out Bottom to Top
iettRandomPoints, // Random Points
iettRandomBoxes, // Random Boxes
iettCenter1, // Wipe Out from Center
iettCenter2, // Wipe In to Center
iettCenterZoom1, // Expand Out from
CenteriettCenterZoom2 // Expand In to Center
{$ifdef IEINCLUDEEXTRATRANSITIONS}
,iettExpandFromLeft, // Expand from Left
iettExpandFromRight, // Expand from
```

ImageEnProc

```
RightiettExpandFromTop, // Expand from
TopiettExpandFromBottom, // Expand from Bottom
iettExpandFromTopLeft, // Expand from Top Left
iettExpandFromTopRight, // Expand from Top Right
iettExpandFromBottomLeft, // Expand from Bottom
Left
iettExpandFromBottomRight, // Expand from Bottom
Right
iettExpandInFromLeft, // Expand in from Left
iettExpandInFromRight, // Expand in from Right
iettExpandInFromTop, // Expand in from Top
iettExpandInFromBottom, // Expand in from Bottom
iettExpandInToVerticalCenter, // Expand in to
Vertical Center
iettExpandInToHorizon, // Expand in to Horizon
iettExpandInFromSides, // Expand in from Sides
iettExpandInFromTopAndBottom, // Expand in from
Top and Bottom
iettExpandOutFromHorizon, // Expand out from
Horizon
iettExpandOutFromVerticalCenter, // Expand out
from Vertical Center
iettWipeFromTopLeft, // Wipe from Top Left
iettWipeFromTopRight, // Wipe from Top Right
iettWipeFromBottomLeft, // Wipe from Bottom Left
iettWipeFromBottomRight, // Wipe from Bottom
Right
iettWipeInFromTopAndBottom, // Wipe in from Top
and Bottom
iettWipeFromHorizon, // Wipe from Horizon
iettWipeInFromSides, // Wipe in from Sides
```

```
iettWipeOutFromVerticalCenter, // Wipe out from
Vertical Center
iettBuildUpFromLeft, // Build up from Left
iettBuildUpFromRight, // Build up from Right
iettBuildUpFromTop, // Build up from Top
iettBuildUpFromBottom, // Build up from Bottom
iettUnrollFromLeft, // Unroll from Left
iettUnrollFromRight, // Unroll from Right
iettUnrollFromTop, // Unroll from Top
iettUnrollFromBottom, // Unroll from Bottom
iettSlideInFromLeft, // Slide in from Left
iettSlideInFromRight, // Slide in from
RightiettSlideInFromTop, // Slide in from Top
iettSlideInFromBottom, // Slide in from Bottom
iettShrinkToTopLeft, // Shrink to Top Left
iettShrinkToTopRight, // Shrink to Top Right
iettShrinkToBottomLeft, // Shrink to Bottom Left
iettShrinkToBottomRight, // Shrink to Bottom Right
iettShrinkToCenter, // Shrink to Center
iettQuartersWipeInToCenter, // Quarters Wipe in to
Center
iettQuartersExpandToCenter, // Quarters Expand to
Center
iettQuartersSlideInToCenter, // Quarters Slide in to
Center
iettCurvedWipeFromLeft, // Curved Wipe from Left
iettCurvedWipeFromRight, // Curved Wipe from Right
iettCurvedWipeFromTop, // Curved Wipe from Top
iettCurvedWipeFromBottom, // Curved Wipe from Bottom
iettCurvedWipeFromTopLeft, // Curved Wipe from Top
Left
iettCurvedWipeFromTopRight, // Curved Wipe from Top
Right
```

ImageEnProc

```
iettCurvedWipeFromBottomLeft, // Curved Wipe from
Bottom Left
iettCurvedWipeFromBottomRight, // Curved Wipe from
Bottom Right
iettBarsInFromLeft, // Bars in from
LeftiettBarsInFromRight, // Bars in from Right
iettBarsFromTop, // Bars from Top


iettBarsFromBottom, // Bars from Bottom
iettBarsLeftThenRight, // Bars Left then Right
iettBarsRightThenLeft, // Bars Right then Left
iettBarsTopThenBottom, // Bars Top then Bottom
iettBarsBottomThenTop, // Bars Bottom then Top
iettBarsFrombothSides, // Bars from both Sides
iettBarsFromTopAndBottom, // Bars from Top and Bottom
iettShreddedFromLeft, // Shredded from Left
iettShreddedFromRight, // Shredded from Right
iettShreddedFromTop, // Shredded from Top
iettShreddedFromBottom, // Shredded from Bottom
iettShreddedFromTopAndLeft, // Shredded from Top and
Left
iettShreddedFromTopAndRight, // Shredded from Top and
Right
iettShreddedFromBottomAndLeft, // Shredded from
Bottom and Left
iettShreddedFromBottomAndRight, // Shredded from
Bottom and Right
iettShreddedFromHorizonAndLeft, // Shredded from
Horizon and Left
iettShreddedFromHorizonAndRight, // Shredded from
Horizon and Right
iettShreddedFromTopAndVerticalCenter, // Shredded
from Top and Vertical Center
iettShreddedFromBottomAndVerticalCenter, // Shredded
from Bottom and Vertical Center
```

```
iettShreddedFromCenter, // Shredded from Center
iettShreddedToCenter, // Shredded to Center
iettShreddedInToHorizon, // Shredded in to Horizon
iettShreddedInToVerticalCenter, // Shredded in to
Vertical Center
iettShreddedOutFromHorizon, // Shredded out from
Horizon
iettShreddedOutFromVerticalCenter, // Shredded out
from Vertical Center
iettExpandingRectangles, // Expanding Rectangles
iettExpandingTriangles, // Expanding Triangles
iettExpandingCircles, // Expanding Circles
iettExpandingDiamonds, // Expanding Diamonds
iettCircularWipeFromCenter, // Circular Wipe from
Center
iettCircularWipeToCenter, // Circular Wipe to Center
iettCrisscrossWipeFromTopLeft, // Crisscross Wipe
from Top Left
iettCrisscrossWipeFromTopRight, // Crisscross Wipe
from Top Right
iettCrisscrossWipeFromBottomLeft, // Crisscross Wipe
from Bottom Left
iettCrisscrossWipeFromBottomRight, // Crisscross Wipe
from Bottom Right
iettCrisscrossWipeBounceFromTopLeft, // Crisscross
Wipe Bounce from Top Left
iettCrisscrossWipeBounceFromTopRight, // Crisscross
Wipe Bounce from Top Right
iettCrisscrossWipeBounceFromBottomLeft, // Crisscross
Wipe Bounce from Bottom Left
iettCrisscrossWipeBounceFromBottomRight, //
Crisscross Wipe Bounce from Bottom Right
iettCrisscrossWipeFromLeftRightAndTop, // Crisscross
Wipe from Left Right and Top
```

ImageEnProc

```
iettCrisscrossWipeFromLeftRightAndBottom, //
Crisscross Wipe from Left Right and Bottom
iettCrisscrossWipeFromLeftTopAndBottom, // Crisscross
Wipe from Left Top and Bottom
iettCrisscrossWipeFromTopLeftRightAndBottom,//
Crisscross Wipe from Top Left Right and Bottom
iettCrisscrossWipeFromRightTopAndBottom, //
Crisscross Wipe from Right Top and Bottom
iettCrisscrossWipeFromBottomLeftTopRight, //
Crisscross Wipe from Bottom Left Top Right
iettWipeDiagonalFromTopLeft, // Wipe diagonal from
Top Left
iettWipeDiagonalFromTopRight, // Wipe diagonal from
Top Right
iettWipeDiagonalFromBottomLeft, // Wipe diagonal from
Bottom Left
iettWipeDiagonalFromBottomRight, // Wipe diagonal
from Bottom Right
iettDiagonalSweepClockwise, // Diagonal Sweep
Clockwise
iettDiagonalSweepCounterClockwise, // Diagonal Sweep
Counter-Clockwise
iettSweepClockwise, // Sweep Clockwise
iettSweepCounterClockwise, // Sweep Counter-Clockwise
iettStarburstClockwiseFromCenter, // Starburst
Clockwise from Center
iettStarburstCounterClockwiseFromCenter, // Starburst
Clockwise from Center
iettRotationalRectangle, // Rotational Rectangle –
Clockwise
iettRotationalRectangleCounterClockwise, //
Rotational Rectangle - Counter Clockwise
iettRotationalStar, // Rotational Star – Clockwise
```

ImageEnProc

```
iettRotationalStarCounterClockwise, // Rotational
Star - Counter Clockwise
iettSpeckledWipeFromLeft, // Speckled Wipe from Left
iettSpeckledWipeFromRight, // Speckled Wipe from
RightiettSpeckledWipeFromTop, // Speckled Wipe from
Top
iettSpeckledWipeFromBottom, // Speckled Wipe from
Bottom


iettPushLeftAndSlideOut, // Push Left and Slide out
iettPushRightAndSlideOut, // Push Right and Slide out
iettPushUpAndSlideOut, // Push up and Slide out
iettPushDownAndSlideOut, // Push down and Slide out
iettPushAndSqueezeLeft, // Push and Squeeze Left
iettPushAndSqueezeRight, // Push and Squeeze Right
iettPushAndSqueezeUp, // Push and Squeeze up
iettPushAndSqueezeDown, // Push and Squeeze down
iettHorizontalBlinds, // Horizontal Blinds
iettVerticalBlinds, // Vertical Blinds
iettUnevenBlindsFromLeft, // Uneven Blinds from Left
iettUnevenBlindsFromRight, // Uneven Blinds from
Right
iettUnevenBlindsFromTop, // Uneven Blinds from Top
iettUnevenBlindsFromBottom, // Uneven Blinds from
Bottom
iettRectanglesFromTheLeft, // Rectangles from the
Left
iettRectanglesFromTheRight, // Rectangles from the
Right
iettRectanglesFromTheTop, // Rectangles from the Top
iettRectanglesFromTheBottom, // Rectangles from the
Bottom
iettSpirallingRectangleClockwise, // Spiralling
Rectangle Clockwise
```

ImageEnProc

```
iettSpirallingRectangleCounterClockwise, //
Spiralling Rectangle Counter-Clockwise
iettArrowWipeFromLeft, // Arrow Wipe from Left
iettArrowWipeFromRight, // Arrow Wipe from Right
iettArrowWipeFromTop, // Arrow Wipe from Top
iettArrowWipeFromBottom, // Arrow Wipe from Bottom
iettHorizontalBowTieWipe, // Horizontal Bow Tie Wipe
iettVerticalBowTieWipe, // Vertical Bow Tie Wipe
iettDiagonalCrossFromCenter, // Diagonal Cross from
Center
iettDiagonalCrossToCenter, // Diagonal Cross to
Center
iettZigzagWipeFromHorizon, // Zigzag Wipe from
Horizon
iettZigzagWipeFromVerticalCenter, // Zigzag Wipe from
Vertical Center
iettDiamondWipeFromCenter, // Diamond Wipe from
Center
iettDiamondWipeToCenter, // Diamond Wipe to Center
iettDiagonalBoxWipe, // Diagonal Box Wipe
iettTriangularWipeEffect, // Triangular Wipe
iettRandomBigBoxesEffect, // Random Big Boxes
iettPageFlipEffect, // Page Flip
iettPageFlipEffect2, // Page Flip 2
iettReversePageFlip, // Page Flip
iettReversePageFlip2, // Page Flip 2
iettZigzagWipeToHorizon, // Zigzag Wipe To Horizon
iettZigzagWipeToVerticalCenter, // Zigzag Wipe To
Vertical Center
iettRandomHearts, // Random Hearts
iettRandomStar5s, // Random 5 Pointed Stars
iettRandomStar6s, // Random 6 Pointed Stars
iettRandomExplosions, // Random Explosions
iettExpandingHearts, // Expanding Hearts
iettExpandingStar5, // Expanding 5 Pointed Stars
```

ImageEnProc

```
iettExpandingStar6, // Expanding 6 Pointed
StarsiettExpandingExplosions, // Expanding Explosions
iettExpandingLightningBolts, // Expanding Lightning
Bolts
iettHeartWipeOut, // Heart Wipe from Center
iettHeartWipeIn, // Heart Wipe to Center
iettStar5WipeOut, // 5 Pointed Star Wipe from Center
iettStar5WipeIn, // 5 Pointed Star Wipe to Center
iettStar6WipeOut, // 6 Pointed Star Wipe from Center
iettStar6WipeIn, // 6 Pointed Star Wipe to Center
iettExplosionWipeOut, // Explosion Wipe from Center
iettExplosionWipeIn, // Explosion Wipe to Center
iettCrossWipeOut, // Cross Wipe from Center
iettCrossWipeIn, // Cross Wipe to Center
iettHeartWipeInAndOut, // Heart Wipe In and Out
iettStar5WipeInAndOut, // 5 Pointed Star Wipe In and
Out
iettStar6WipeInAndOut, // 6 Pointed Star Wipe In and
Out
iettExplosionWipeInAndOut, // Explosion Wipe In and
Out
iettCubeRotateFromLeft, // Cube Rotate from Left
iettCubeRotateFromRight, // Cube Rotate from Right
iettCubeRotateFromTop, // Cube Rotate from Top
iettCubeRotateFromBottom, // Cube Rotate from Bottom
iettSoftWipeFromLeft, // Soft Wipe from Left
iettSoftWipeFromRight, // Soft Wipe from Right
iettSoftWipeFromTop, // Soft Wipe from Top
iettSoftWipeFromBottom, // Soft Wipe from Bottom
iettAngledTwistIn, // Twist In
iettAngledTwistOut, // Twist Out
iettMultipleAngledTwistIn, // Multiple Twist In
iettMultipleAngledTwistOut // Multiple Twist
Out{$endif}
,iettPanZoom
);
```

Unit ImageENProc

TImageEnProc

Declaration

```
TResampleFilter = (rfNone, rfTriangle, rfHermite,
rfBell, rfBSpline, rfLanczos3, rfMitchell,
rfNearest, rfLinear, rfFastLinear, rfBilinear,
rfBicubic, rfProjectBW, rfProjectWB);
```

Description

If you need the best quality we suggest: rfHermite, rfBell, rfBSpline, rfLanczos3, rfMitchell, rfNearest, rfBilinear, rfBicubic. If you need speed we suggest: rfTriangle, rfLinear, rfFastLinear.

For projects (white on black or black on white) we suggest: rfProjectBW and rfProjectWB

Others

DECLARATION

```
property AutoConvertFormat: Boolean;
```

DESCRIPTION

When true (default) all image processing functions converts automatically the source pixel format to the request one. For example, Contrast method requires RGB24 pixel format. If your image is black/white or any other format, it is converted automatically to RGB24. If you set AutoConvertFormat := False and call Contrast with a black/white image, contrast just will don't work.

DECLARATION

```
property Background: TColor;
```

DESCRIPTION

This property specifies the background color. The background color is the color shown in the unoccupied area when the current image is less than the control's size. This color is used also in geometric processing (such as rotation) to fill blank areas. When TImageEnProc is attached to TImageEnView, the TImageEnProc.Background is equal to TImageEnView.Background.

Unit ImageENProc

TImageEnProc

Declaration

```
constructor Create(Owner: TComponent);
```

Description

Create creates a new instance. You can set Owner=nil to create a component without owner.

Declaration

```
constructor CreateFromBitmap(Bitmap: TIEBitmap);
constructor CreateFromBitmap(Bitmap: TBitmap);
```

Description

CreateFromBitmap creates a new instance assigning property AttachedIEBitmap or AttachedBitmap. CreateFromBitmap sets AutoUndo to false.

ImageEnProc

Declaration

```
function GetReSel(var fsX1, fsY1, fsX2,
fsY2:integer; var PolySel: PPointArray; var
PolySelCount: integer; var mask: TIEMask):
Boolean;
```

Description

The GetReSel function returns the selected area. Returns False if there isn't a selected area.

Unit ImageENProc
TImageEnProc

DECLARATION

```
procedure Update;
```

DESCRIPTION

If TImageEnProc is attached to a TImageEnView (or inherited) object, Update calls relative TImageEnView.Update method. If TImageEnProc is attached to a TBitmap object, update sets the modified property to True.

ImageEnProc

Events

DECLARATION

```
property OnFinishWork: TNotifyEvent;
```

DESCRIPTION

This event occurs whenever an image processing task terminates. This event is called after the last call to OnProgress, so you can reset here your progress bar.

DECLARATION

```
property OnPreview: TIEPreviewEvent;
```

DESCRIPTION

The OnPreview event is called before the preview form is displayed (running DoPreviews method).

DECLARATION

```
property OnProgress: TIEProgressEvent;
```

DESCRIPTION

The OnProgress event is called when image processing operations are executed.

Unit ImageENProc

TImageEnProc

DECLARATION

```
property OnSaveUndo: TIESaveUndoEvent;
```

DESCRIPTION

OnSaveUndo is called after SaveUndo is called by user action or by code. Source is the same parameter used calling SaveUndo.

Chapter 18

TImageEnVideoView

Chapter 18. ImageEnVideoView

TImageEnVideoView is derived directly from TImageEnVect. TImageEnVideoView has some properties and methods (zoom, scroll-bars... and, above all, bitmap field and vectorial object capability).

When you set ShowVideo property to true the current image of TImageEnVideoView is overlapped from video input (stretched to current size of TImageEnVideoView component).

We suggest using TImageEnView.IO.DShowParams in order to capture video from cameras, because this one uses the more supported DirectShow API.

Unit ImageEnVidoView

TImageEnVideoView

Methods and Properties

AudioBitsPerSample	AudioChannels	AudioFormat
AudioSamplesPerSec	DisplayMode	DoConfigureCompression
DoConfigureDisplay	DoConfigureFormat	DoConfigureSource
FitFreeze	Freeze	Frozen
GetVideoSize	HasDlgVideoDisplay	HasDlgVideoFormat
HasDlgVideoSource	HasOverlay	PreviewRate
RecAudio	RecFileName	RecFrameRate
RecMultitask	SaveFrame	ShowVideo
StartRecord	StopRecord	UnFreeze
VideoSource	VideoSourceList	WndCaptureHandle

Events

OnJob	OnVideoFrameRaw	OnVideoFrame

Methods and Properties

DECLARATION

```
property AudioBitsPerSample: word;
```

DESCRIPTION

AudioChannels, AudioSamplesPerSec, AudioBitsPerSample and AudioFormat properties allow the application to get/set the audio recording format. AudioBitsPerSample specifies the bits per sample for the AudioFormat format type. If a compression scheme cannot define a bits-per-sample value, this field is zero.

DECLARATION

```
property AudioChannels: word;
```

DESCRIPTION

AudioChannels, AudioSamplesPerSec, AudioBitsPerSample and AudioFormat properties allow the application to get/set the audio recording format. AudioChannels specifies the number of channels in the waveform-audio data. Monaural data uses one channel and stereo data uses two channels.

Unit ImageEnVidoView

TImageEnVideoView

DECLARATION

```
property AudioFormat: word;
```

DESCRIPTION

AudioChannels, AudioSamplesPerSec, AudioBitsPerSample and AudioFormat properties allow the application to get/set the audio recording format. AudioFormat specifies the compression audio format. Currently defined values are:

$0000 : UNKNOWN
$0001 : PCM
$0002 : ADPCM
$0003 : IEEE_FLOAT
$0004 : VSELP
$0005 : IBM_CVSD
$0006 : ALAW
$0007 : MULAW
$0008 : DTS
$0010 : OKI_ADPCM
$0011 : DVI_ADPCM
$0012 : MEDIASPACE_ADPCM
$0013 : SIERRA_ADPCM
$0014 : G723_ADPCM
$0015 : DIGISTD
$0016 : DIGIFIX
$0017 : DIALOGIC_OKI_ADPCM
$0018 : MEDIAVISION_ADPCM
$0019 : CU_CODEC
$0020 : YAMAHA_ADPCM
$0021 : SONARC

$0022 : DSPGROUP_TRUESPEECH
$0023 : ECHOSC1
$0024 : AUDIOFILE_AF36
$0025 : APTX
$0026 : AUDIOFILE_AF10
$0027 : PROSODY_1612
$0028 : LRC
$0030 : DOLBY_AC2
$0031 : GSM610
$0032 : MSNAUDIO
$0033 : ANTEX_ADPCME
$0034 : CONTROL_RES_VQLPC
$0035 : DIGIREAL
$0036 : DIGIADPCM
$0037 : CONTROL_RES_CR10
$0038 : NMS_VBXADPCM
$0039 : CS_IMAADPCM
$003A : ECHOSC3
$003B : ROCKWELL_ADPCM
$003C : ROCKWELL_DIGITALK
$003D : XEBEC
$0040 : G721_ADPCM
$0041 : G728_CELP
$0042 : MSG723
$0050 : MPEG
$0052 : RT24
$0053 : PAC

$0055 : MPEGLAYER3

$0059 : LUCENT_G723
$0060 : CIRRUS
$0061 : ESPCM
$0062 : VOXWARE
$0063 : CANOPUS_ATRAC
$0064 : G726_ADPCM
$0065 : G722_ADPCM
$0067 : DSAT_DISPLAY
$0069 : VOXWARE_BYTE_ALIGNED
$0070 : VOXWARE_AC8
$0071 : VOXWARE_AC10
$0072 : VOXWARE_AC16
$0073 : VOXWARE_AC20
$0074 : VOXWARE_RT24
$0075 : VOXWARE_RT29
$0076 : VOXWARE_RT29HW
$0077 : VOXWARE_VR12
$0078 : VOXWARE_VR18
$0079 : VOXWARE_TQ40
$0080 : SOFTSOUND
$0081 : VOXWARE_TQ60
$0082 : MSRT24
$0083 : G729A
$0084 : MVI_MVI2
$0085 : DF_G726
$0086 : DF_GSM610
$0088 : ISIAUDIO

$0089 : ONLIVE
$0091 : SBC24
$0092 : DOLBY_AC3_SPDIF
$0093 : MEDIASONIC_G723
$0094 : PROSODY_8KBPS
$0094 : ZYXEL_ADPCM
$0098 : PHILIPS_LPCBB
$0099 : PACKED
$00A0 : MALDEN_PHONYTALK
$0100 : RHETOREX_ADPCM
$0101 : IRAT
$0111 : VIVO_G723
$0112 : VIVO_SIREN
$0123 : DIGITAL_G723
$0125 : SANYO_LD_ADPCM
$0130 : SIPROLAB_ACEPLNET
$0131 : SIPROLAB_ACELP4800
$0132 : SIPROLAB_ACELP8V3
$0133 : SIPROLAB_G729
$0134 : SIPROLAB_G729A
$0135 : SIPROLAB_KELVIN
$0140 : G726ADPCM
$0150 : QUALCOMM_PUREVOICE
$0151 : QUALCOMM_HALFRATE
$0155 : TUBGSM
$0160 : MSAUDIO1
$0200 : CREATIVE_ADPCM
$0202 : CREATIVE_FASTSPEECH8
$0203 : CREATIVE_FASTSPEECH10
$0210 : UHER_ADPCM
$0220 : QUARTERDECK

$0230 : ILINK_VC
$0240 : RAW_SPORT
$0250 : IPI_HSX
$0251 : IPI_RPELP
$0260 : CS2
$0270 : SONY_SCX
$0300 : FM_TOWNS_SND
$0400 : BTV_DIGITAL
$0450 : QDESIGN_MUSIC
$0680 : VME_VMPCM
$0681 : TPC
$1000 : OLIGSM
$1001 : OLIADPCM
$1002 : OLICELP
$1003 : OLISBC
$1004 : OLIOPR
$1100 : LH_CODEC
$1400 : NORRIS
$1500 : SOUNDSPACE_MUSICOMPRESS
$2000 : DVM

DECLARATION

```
property AudioSamplesPerSec: dword;
```

DESCRIPTION

AudioChannels, AudioSamplesPerSec, AudioBitsPerSample and AudioFormat properties allow the application to get/set the audio recording format. AudioSamplesPerSec specifies the sampling rate, in samples per second (hertz), at which each channel should be played or recorded.

DECLARATION

```
property DisplayMode: TVCDisplayMode;
```

DESCRIPTION

Select Overlay (dmOverlay) or Preview (dmPreview) display mode. The default is dmPreview.

DECLARATION

```
function DoConfigureCompression: Boolean;
```

DESCRIPTION

DoConfigureCompression executes the Configure Compression Dialog of the selected driver (see VideoSource property). If the driver is busy (fails to open), the function returns False.

Unit ImageEnVidoView

TImageEnVideoView

DECLARATION

```
function DoConfigureDisplay: Boolean;
```

DESCRIPTION

DoConfigureDisplay executes the Configure Display Dialog of the selected driver (see VideoSource property). If the driver is busy or it has failed to open, the function returns False.

DECLARATION

```
function DoConfigureFormat: Boolean;
```

DESCRIPTION

DoConfigureFormat executes the Configure Format Dialog of the selected driver (see VideoSource property). If the driver is busy or it has failed to open, the function returns False.

DECLARATION

```
function DoConfigureSource: Boolean;
```

DESCRIPTION

DoConfigureSource executes the Configure Source Dialog of the selected driver (see VideoSource property). If the driver is busy or it has failed to open, the function returns False.

Unit ImageEnVidoView

TImageEnVideoView

DECLARATION

```
property FitFreeze: Boolean;
```

DESCRIPTION

If True (default) the video input frozen and is adapted to the component size (uses a triangular filter if the video input is smaller than component, to improve quality), otherwise the frozen image is what you have selected with Format dialog.

DECLARATION

```
procedure Freeze;
```

DESCRIPTION

Freeze sets the frozen property to True. When you set the Frozen property to True, the video input is locked and the current image is copied to Bitmap field (in 24 bit x pixel). If you want to process/zoom/navigate the captured image, you must set the ShowVideo property to False.

The size of image is equal to size of TImageEnVideoView component (it is your responsibility to maintain the correct aspect ratio) if FitFreeze is True.

Unit ImageEnVidoView

TImageEnVideoView

DECLARATION

```
property Frozen: Boolean;
```

DESCRIPTION

When you set the Frozen property to True, the video input is locked and the current image is copied to Bitmap field (in 24 bit x pixel). If you want process/zoom/navigate the image you have to set ShowVideo property to False.

The size of image is equal to size of TImageEnVideoView component (is your responsibility to maintain the correct aspect ratio) if FitFreeze is True.

DECLARATION

```
function GetVideoSize: TRect;
```

DESCRIPTION

GetVideoSize returns the rectangle of video input (as selected from ConfigureFormat dialog).

ImageEnVideoView

Unit ImageEnVidoView

TImageEnVideoView

DECLARATION

```
property HasDlgVideoDisplay: Boolean;
```

DESCRIPTION

HasDlgVideoDisplay returns True if the selected driver supports a Video Display Dialog.

Read-only

DECLARATION

```
property HasDlgVideoFormat: Boolean;
```

DESCRIPTION

HasDlgVideoFormat returns true if the selected driver supports a Video Format Dialog.

Read-only

ImageEnVideoView

Unit ImageEnVidoView

TImageEnVideoView

DECLARATION

```
property HasDlgVideoSource: Boolean;
```

DESCRIPTION

HasDlgVideoSource returns true if the selected driver supports a Video Source Dialog.

Read-only

DECLARATION

```
property HasOverlay: Boolean;
```

DESCRIPTION

HasOverlay returns true if the selected driver supports Overlay display mode.

Read-only

DECLARATION

```
property PreviewRate: integer;
```

DESCRIPTION

The PreviewRate sets the interval (in milliseconds) between the acquisition of successive frames. It is valid only if DisplayMode is dmPreview.

Unit ImageEnVidoView

TImageEnVideoView

DECLARATION

```
property RecAudio: boolean;
```

DESCRIPTION

Set RecAudio to True to capture audio input with video input. The default is False.

DECLARATION

```
property RecFileName: AnsiString;
```

DESCRIPTION

RecFileName contains the file name (AVI file format) where to save the captured video input. Default is 'Capture.avi'.

DECLARATION

```
property RecFrameRate: integer;
```

DESCRIPTION

RecFrameRate is the number of frames per second captured on recording. Default is 15.

Unit ImageEnVidoView
TImageEnVideoView

DECLARATION

```
property RecMultitask: Boolean;
```

DESCRIPTION

If RecMultitask is False the system is locked to wait the end of recording. Stop recording by pressing the ESC key. Default is true.

DECLARATION

```
procedure SaveFrame;
```

DESCRIPTION

SaveFrame saves current frame without locking (see freeze) the video input. Application can display the image in the component's visual area by setting ShowVideo to False.

DECLARATION

```
property ShowVideo: Boolean;
```

DESCRIPTION

Set ShowVideo to True to show the current video source input (see VideoSource property) and hides the current image of TImageEnVideoView component. The scrollbars will be hidden.

ImageEnVideoView

You can set this property to True at design time and see the video input, but remember to set it to False when running the application, because only one video input is allowed at a time. You can have multiple TImageEnVideoView component on your form but only one can have ShowVideo property set to True (if they use some video source).

DECLARATION

```
function StartRecord: Boolean;
```

DESCRIPTION

Begin recording of the video input to AVI format. To select compression algorithm run the ConfigureCompression dialog. StartRecord returns false if it fails, true if it's successful.

DECLARATION

```
procedure StopRecord;
```

DESCRIPTION

Stop recording begun with StartRecord. After StopRecord completes, you can access the saved AVI file.

Unit ImageEnVidoView

TImageEnVideoView

Declaration

```
procedure UnFreeze;
```

Description

UnFreeze sets the Frozen property to False. And Unlocks video input. The Bitmap field of TImageEnVideoView contains the last frozen image.

Declaration

```
property VideoSource: integer;
```

Description

VideoSource property contains the index of the current video source (see VideoSourceList). The default is 0.

Declaration

```
property VideoSourceList: TStringList;
```

Description

This is the list of video capture (video source) drivers installed on the system. The format is: “Device_Name Device_Version”. The index of the list corresponds to VideoSource property value.

Read-only

DECLARATION

```
property WndCaptureHandle: HWND;
```

DESCRIPTION

WndCaptureHandle is the handle of the video capture window. It is useful to send messages to Video for Windows system.

Events

DECLARATION

```
property OnJob: TIEJobEvent;
```

DESCRIPTION

The OnJob event is generated on video capture jobs, as connecting, video format negotiations... Supported job of TImageEnVideoView have "iejVIDEOCAP_" prefix. See TIEJobEvent type for more details.

DECLARATION

```
property OnVideoFrameRaw: TVideoFrameRawEvent;
```

DESCRIPTION

This event is generated for each input frame (as OnVideoFrame). The TVideoFrameRawEvent function is defined in this way:
You can modify the pixels (pData) because this event is generated before video source shows the frame.

Unit ImageEnVidoView

TImageEnVideoView

DECLARATION

```
property OnVideoFrame: TVideoFrameEvent;
```

DESCRIPTION

This event is generated for each input frame. If you handle this event, the performance of video input degrades. You haven't to free this bitmap: ImageEn will free it.

Chapter 19

TImageEnVideoCap

Chapter 19. ImageEnVideoCap

TImageEnVideoCap is a non-visual component which can capture images from video cameras. It uses VFW (Video for Windows) to capture frames from a video source.

We suggest using TImageEnView.IO.DShowParams in order to capture video from cameras, because this one uses the more supported DirectShow API.

See also

TImageEnVideoView

Unit ImageEnVidoView

TImageEnVideoCap

Methods and Properties

AudioBitsPerSample	AudioChannels	AudioFormat
AudioSamplesPerSec	Capture	DoConfigureCompression
DoConfigureDisplay	DoConfigureFormat	DoConfigureSource
GetVideoSize	HasDlgVideoDisplay	HasDlgVideoFormat
HasDlgVideoSource	HasOverlay	RecAudio
RecFileName	RecFrameRate	RecMultitask
StartRecord	StopRecord	UseWindowsCodec
VideoSource	VideoSourceList	WndCaptureHandle

Events

OnJob	OnVideoFrameRaw	OnVideoFrame

Methods and Properties

DECLARATION

```
PROPERTY AUDIOBITSPERSAMPLE: WORD;
```

DESCRIPTION

AudioChannels, AudioSamplesPerSec, AudioBitsPerSample and AudioFormat properties allow the application to get/set the audio recording format. AudioBitsPerSample specifies the bits per sample for the AudioFormat format type. If a compression scheme cannot define a bits-per-sample value, this field is zero.

DECLARATION

```
property AudioChannels: word;
```

DESCRIPTION

AudioChannels, AudioSamplesPerSec, AudioBitsPerSample and AudioFormat properties allow the application to get/set the audio recording format. AudioChannels specifies the number of channels in the waveform-audio data. Monaural data uses one channel and stereo data uses two channels.

ImageEnVideoCap

DECLARATION

```
property AudioFormat: word;
```

DESCRIPTION

AudioChannels, AudioSamplesPerSec, AudioBitsPerSample and AudioFormat properties allow the application to get/set the audio recording format. AudioFormat specifies the compression audio format. Currently defined values are:
$0000 : UNKNOWN
$0001 : PCM
$0002 : ADPCM
$0003 : IEEE_FLOAT
$0004 : VSELP
$0005 : IBM_CVSD
$0006 : ALAW
$0007 : MULAW
$0008 : DTS
$0010 : OKI_ADPCM
$0011 : DVI_ADPCM
$0012 : MEDIASPACE_ADPCM
$0013 : SIERRA_ADPCM
$0014 : G723_ADPCM

$0015 : DIGISTD
$0016 : DIGIFIX
$0017 : DIALOGIC_OKI_ADPCM
$0018 : MEDIAVISION_ADPCM
$0019 : CU_CODEC
$0020 : YAMAHA_ADPCM
$0021 : SONARC
$0022 : DSPGROUP_TRUESPEECH
$0023 : ECHOSC1
$0024 : AUDIOFILE_AF36
$0025 : APTX
$0026 : AUDIOFILE_AF10
$0027 : PROSODY_1612
$0028 : LRC
$0030 : DOLBY_AC2
$0031 : GSM610
$0032 : MSNAUDIO
$0033 : ANTEX_ADPCME
$0034 : CONTROL_RES_VQLPC
$0035 : DIGIREAL
$0036 : DIGIADPCM
$0037 : CONTROL_RES_CR10
$0038 : NMS_VBXADPCM
$0039 : CS_IMAADPCM
$003A : ECHOSC3
$003B : ROCKWELL_ADPCM
$003C : ROCKWELL_DIGITALK
$003D : XEBEC
$0040 : G721_ADPCM

ImageEnVideoCap

$0041 : G728_CELP
$0042 : MSG723
$0050 : MPEG
$0052 : RT24
$0053 : PAC
$0055 : MPEGLAYER3
$0059 : LUCENT_G723
$0060 : CIRRUS
$0061 : ESPCM
$0062 : VOXWARE
$0063 : CANOPUS_ATRAC
$0064 : G726_ADPCM
$0065 : G722_ADPCM
$0067 : DSAT_DISPLAY
$0069 : VOXWARE_BYTE_ALIGNED
$0070 : VOXWARE_AC8
$0071 : VOXWARE_AC10
$0072 : VOXWARE_AC16
$0073 : VOXWARE_AC20
$0074 : VOXWARE_RT24
$0075 : VOXWARE_RT29
$0076 : VOXWARE_RT29HW
$0077 : VOXWARE_VR12
$0078 : VOXWARE_VR18
$0079 : VOXWARE_TQ40
$0080 : SOFTSOUND
$0081 : VOXWARE_TQ60
$0082 : MSRT24

ImageEnVideoCap

$0083 : G729A
$0084 : MVI_MVI2
$0085 : DF_G726
$0086 : DF_GSM610
$0088 : ISIAUDIO
$0089 : ONLIVE
$0091 : SBC24
$0092 : DOLBY_AC3_SPDIF
$0093 : MEDIASONIC_G723
$0094 : PROSODY_8KBPS
$0094 : ZYXEL_ADPCM
$0098 : PHILIPS_LPCBB
$0099 : PACKED
$00A0 : MALDEN_PHONYTALK
$0100 : RHETOREX_ADPCM
$0101 : IRAT
$0111 : VIVO_G723
$0112 : VIVO_SIREN
$0123 : DIGITAL_G723
$0125 : SANYO_LD_ADPCM
$0130 : SIPROLAB_ACEPLNET
$0131 : SIPROLAB_ACELP4800
$0132 : SIPROLAB_ACELP8V3
$0133 : SIPROLAB_G729
$0134 : SIPROLAB_G729A
$0135 : SIPROLAB_KELVIN
$0140 : G726ADPCM
$0150 : QUALCOMM_PUREVOICE

ImageEnVideoCap

$0151 : QUALCOMM_HALFRATE
$0155 : TUBGSM
$0160 : MSAUDIO1
$0200 : CREATIVE_ADPCM
$0202 : CREATIVE_FASTSPEECH8
$0203 : CREATIVE_FASTSPEECH10
$0210 : UHER_ADPCM
$0220 : QUARTERDECK
$0230 : ILINK_VC
$0240 : RAW_SPORT
$0250 : IPI_HSX
$0251 : IPI_RPELP
$0260 : CS2
$0270 : SONY_SCX
$0300 : FM_TOWNS_SND
$0400 : BTV_DIGITAL
$0450 : QDESIGN_MUSIC
$0680 : VME_VMPCM
$0681 : TPC
$1000 : OLIGSM
$1001 : OLIADPCM
$1002 : OLICELP
$1003 : OLISBC
$1004 : OLIOPR
$1100 : LH_CODEC
$1400 : NORRIS
$1500 : SOUNDSPACE_MUSICOMPRESS
$2000 : DVM

ImageEnVideoCap

DECLARATION

```
property AudioSamplesPerSec: dword;
```

DESCRIPTION

AudioChannels, AudioSamplesPerSec, AudioBitsPerSample and AudioFormat properties allow the application to get/set the audio recording format. AudioSamplesPerSec specifies the sampling rate, in samples per second (hertz), at which each channel should be played or recorded.

DECLARATION

```
property Capture: boolean;
```

DESCRIPTION

Set Capture to True to activate video frames capture. When capture is true, video frames are sent to OnVideoFrame and OnVideoFrameRaw events.

DECLARATION

```
property DisplayMode: TVCDisplayMode;
```

DESCRIPTION

Select Overlay (dmOverlay) or Preview (dmPreview) display mode. The default is dmPreview.

DECLARATION

```
function DoConfigureCompression: Boolean;
```

DESCRIPTION

DoConfigureCompression executes the Configure Compression Dialog of the selected driver (see VideoSource property). If the driver is busy (fails to open), the function returns False.

DECLARATION

```
function DoConfigureDisplay: Boolean;
```

DESCRIPTION

DoConfigureDisplay executes the Configure Display Dialog of the selected driver (see VideoSource property). If the driver is busy or it has failed to open, the function returns False.

DECLARATION

```
function DoConfigureFormat: Boolean;
```

DESCRIPTION

DoConfigureFormat executes the Configure Format Dialog of the selected driver (see VideoSource property). If the driver is busy or it has failed to open, the function returns False.

ImageEnVideoCap

Unit ImageEnVidoView
TImageEnVideoCap

DECLARATION

```
function DoConfigureSource: Boolean;
```

DESCRIPTION

DoConfigureSource executes the Configure Source Dialog of the selected driver (see VideoSource property). If the driver is busy or it has failed to open, the function returns False.

DECLARATION

```
property FitFreeze: Boolean;
```

DESCRIPTION

If FitFreeze is true (default) the video input frozen and is adapted to the component size (uses a triangular filter if the video input is smaller than component, to improve quality), otherwise the frozen image is what you have selected with Format dialog.

Unit ImageEnVidoView
TImageEnVideoCap

DECLARATION

```
procedure Freeze;
```

DESCRIPTION

Freeze sets the frozen property to true. When you set the Frozen property to True, the video input is locked and the current image is copied to Bitmap field (in 24 bit x pixel). If you want to process/zoom/navigate the captured image, you must set the ShowVideo property to False.

The size of image is equal to size of TImageEnVideoView component (it is your responsibility to maintain the correct aspect ratio) if FitFreeze is True.

DECLARATION

```
property Frozen: Boolean;
```

DESCRIPTION

When you set the frozen property to true, the video input is locked and the current image is copied to bitmap field (in 24 bit x pixel). If you want process/zoom/navigate the image you have to set ShowVideo property to false.

The size of image is equal to size of TImageEnVideoView component (is your responsibility to maintain the correct aspect ratio) if FitFreeze is True.

ImageEnVideoCap

DECLARATION

```
function GetVideoSize: TRect;
```

DESCRIPTION

GetVideoSize returns the rectangle of video input (as selected from ConfigureFormat dialog).

DECLARATION

```
property HasDlgVideoDisplay: Boolean;
```

DESCRIPTION

HasDlgVideoDisplay returns true if the selected driver supports a Video Display Dialog.

Read-only

DECLARATION

```
property HasDlgVideoFormat: Boolean;
```

DESCRIPTION

HasDlgVideoFormat returns true if the selected driver supports a Video Format Dialog.

Read-only

Unit ImageEnVidoView

TImageEnVideoCap

DECLARATION

```
property HasDlgVideoSource: Boolean;
```

DESCRIPTION

HasDlgVideoSource returns true if the selected driver supports a Video Source Dialog.

Read-only

DECLARATION

```
property HasOverlay: Boolean;
```

DESCRIPTION

HasOverlay returns true if the selected driver supports Overlay display mode.

Read-only

DECLARATION

```
property PreviewRate: integer;
```

DESCRIPTION

The PreviewRate sets the interval (in milliseconds) between the acquisition of successive frames. It is valid only if DisplayMode is dmPreview.

DECLARATION

```
property RecFileName: AnsiString;
```

DESCRIPTION

RecFileName contains the file name (AVI file format) where to save the captured video input. Default is 'Capture.avi'.

DECLARATION

```
property RecFrameRate: integer;
```

DESCRIPTION

RecFrameRate is the number of frames per second captured on recording. Default is 15.

Unit ImageEnVidoView
TImageEnVideoCap

DECLARATION

```
property RecMultitask: Boolean;
```

DESCRIPTION

If RecMultitask is False the system is locked to wait the end of recording. Stop recording by pressing the ESC key. Default is true.

DECLARATION

```
procedure SaveFrame;
```

DESCRIPTION

SaveFrame saves current frame without locking (see freeze) the video input. Application can display the image in the component's visual area by setting ShowVideo to False.

DECLARATION

```
property ShowVideo: Boolean;
```

DESCRIPTION

Set ShowVideo to True to show the current video source input (see VideoSource property) and hide the current image of TImageEnVideoView component. The scrollbars will be hidden.

ImageEnVideoCap

You can set this property to True at design time and see the video input, but remember to set it to False when running the application, because only one video input is allowed at a time. You can have multiple TImageEnVideoView component on your form but only one can have ShowVideo property set to True (if they use some video source).

DECLARATION

```
function StartRecord: Boolean;
```

DESCRIPTION

Begin recording of the video input to AVI format. To select compression algorithm run the ConfigureCompression dialog.

StartRecord returns false if it fails, true if it's successful.

DECLARATION

```
procedure StopRecord;
```

DESCRIPTION

Stop recording begun with StartRecord. After StopRecord completes, you can access the saved AVI file.

Unit ImageEnVidoView
TImageEnVideoCap

DECLARATION

```
property UseWindowsCodec: boolean;
```

DESCRIPTION

If UseWindowsCodec is true, TImageEnVideoCap uses Windows codec to render video stream, otherwise it uses internal codec. The default is True (Windows codecs). Try setting this property to false if you have problems with video capture.

DECLARATION

```
procedure UnFreeze;
```

DESCRIPTION

UnFreeze sets the frozen property to False. Unlocks video input. The Bitmap field of TImageEnVideoView contains the last frozen image.

DECLARATION

```
property VideoSource: integer;
```

DESCRIPTION

VideoSource property contains the index of the current video source (see VideoSourceList). The default is 0.

ImageEnVideoCap

DECLARATION

```
property VideoSourceList: TStringList;
```

DESCRIPTION

This is the list of video capture (video source) drivers installed on the system. The format is: “Device_Name Device_Version”. The index of the list corresponds to VideoSource property value.

Read-only

DECLARATION

```
property WndCaptureHandle: HWND;
```

DESCRIPTION

WndCaptureHandle is the handle of the video capture window. It is useful to send messages to Video for Windows system.

Events

DECLARATION

```
property OnJob: TIEJobEvent;
```

DESCRIPTION

The OnJob event is generated on video capture jobs, as connecting, video format negotiations... Supported job of TImageEnVideoView have “iejVIDEOCAP_” prefix. See TIEJobEvent type for more details.

Unit ImageEnVidoView

TImageEnVideoCap

DECLARATION

```
property OnVideoFrameRaw: TVideoFrameRawEvent;
```

DESCRIPTION

This event is generated for each input frame (as OnVideoFrame). The TVideoFrameRawEvent function is defined in this way:
You can modify the pixels (pData) because this event is generated before video source shows the frame.

DECLARATION

```
property OnVideoFrame: TVideoFrameEvent;
```

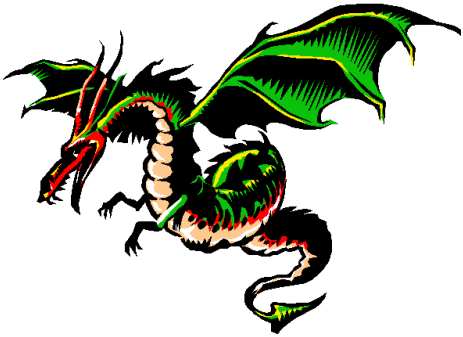
DESCRIPTION

This event is generated for each input frame. If you handle this event, the performance of video input degrades. You haven't to free this bitmap: ImageEn will free it.

Chapter 20

TIEDirectShow

Chapter 20. IEDirectShow



TIEDirectShow class allows control of some Direct Show features, such as video capture, audio capture; multimedia files capture as well video rendering, and multimedia file writing.

Unit IEDS
TIEDirectShow

Methods and Properties

AudioCodecs	AudioInputs	BufferToTIEBitmap
ClockErrorTolerance	Connect	Connected
ConvertTimeFormat	Disconnect	Duration
DVDActivateButton	DVDGetProperty	DVDInputPath
DVDPlayAdvanced	DVDPlayAt	DVDSelectAt
DVDShowMenu	EnableSampleGrabber	EndOfStream
FileInput	FileOutput	GetAverageTimePerFr
GetCurrentVideoForm	GetEventCode	GetSample
GetSupportedTVStand	GetVideoRenderNative	GetVideoRenderRect
NotifyWindow	Pause	Position
Rate	ReferenceClock	RenderAudio
RenderVideo	RepaintVideo	Run
SaveGraph	SetAudioCodec	SetAudioInput
SetCurrentVideoForma	SetNotifyWindow	SetTVStandard
SetVCRHorizontalLocki	SetVideoCodec	SetVideoInput
SetVideoRenderRect	ShowPropertyPages	State
Step	Stop	TimeFormat
TunerChannel	TunerFindSignal	VideoCodecQuality
VideoCodecs	VideoFormats	VideoFormatsCount
VideoInputs	VideoInputSource	VideoInputSources

Methods and Properties

DECLARATION

```
property AudioCodecs: TStringList;
```

DESCRIPTION

AudioCodecs contains a list of audio compression codecs available.

DECLARATION

```
property AudioInputs: TStringList;
```

DESCRIPTION

AudioInputs contains a list of audio inputs available.

DECLARATION

```
procedure BufferToTIEBitmap(buffer: pbyte; len:
integer; DestBitmap: TIEBitmap);
```

DESCRIPTION

For internal use only. Used only if you use TIEDirectShow as standalone object.

DECLARATION

```
property ClockErrorTolerance: integer;
```

DESCRIPTION

ClockErrorTolerance sets the maximum tolerance, in milliseconds, of the audio renderer. The value must be from 1 to 1000, inclusive.

DECLARATION

```
procedure Connect;
```

DESCRIPTION

Connect connects to the specified video input or multimedia file. This makes a DirectShow graph which needs to be activated with Run in order to make it active. Several methods and properties work only after Connect has been successfully called.

IEDirectShow

DECLARATION

```
property Connected: Boolean;
```

DESCRIPTION

Returns true when connected to the video input or multimedia file.

DECLARATION

```
function ConvertTimeFormat(source:int64;
sourceFormat: TIETimeFormat; targetFormat:
TIETimeFormat):int64;
```

DESCRIPTION

ConvertTimeFormat converts from a time format to another one.

DECLARATION

```
procedure Disconnect;
```

DESCRIPTION

Disconnect disconnects from the specified video input or multimedia file.

IEDirectShow

DECLARATION

```
property Duration: int64;
```

DESCRIPTION

Duration returns the duration of the stream (multimedia file) in a format specified by the timeformat property. Duration is measured in 100-nanosecond per unit (when time is selected).

DECLARATION

```
procedure DVDActivateButton;
```

DESCRIPTION

Press a button in DVD menu. To select a position call DVDSelectAt. In order to play DVD inside Delphi IDE please disable “Integrated Debugging”.

DECLARATION

```
function DVDGetProperty(Prop: AnsiString;
SubProp: AnsiString):AnsiString;
```

DESCRIPTION

Read a property from current DVD. This method can return current DVD position and some other properties like number of titles and chapters.

Value	Description
NumOfVolumes	Returns number of volumes
Volume	Returns current used volume
Side	Returns current side (1 or 2)
NumOfTitles	Returns number of titles
DiscID	Returns system-generated 64-bit “unique” identification number for the specified DVD
NumOfChapters	Returns number of chapters of the specified title (specify title in SubProp parameter)
Title	Returns current title
Chapter	Returns current chapter
Time	Returns current time using format like “HH:MM:SS:FF”, where FF is frame in current second.
TotalTitleTime	Retrieves the total playback time for the current title.

DECLARATION

```
property DVDInputPath: WideString;
```

DESCRIPTION

DVDInputPath specifies the driver letter of DVD reader. You can set 'Default' to select default DVD reader. You can also specify a path to VIDEO_TS directory, which contains video data. For example, "C:\mydvd\VIDEO_TS". In order to play DVD inside Delphi IDE please disable "Integrated Debugging".

DECLARATION

```
procedure DVDPlayAdvanced(PlayForward: Boolean;
Speed: double);
```

DESCRIPTION

Change the speed and direction of playing.
Setting PlayForward=true the DVD plays forward at the specified speed from the current location. Setting playForward=false the DVD plays backward at the specified speed from the current location.

Speed specifies the playback speed. This value is a multiplier, where 1.0 is the authored speed, so a value of 2.5 plays at two and one-half times the authored speed, while a value of 0.5 plays at half the authored speed. The actual speed of playback depends on the capabilities of the video decoder. Values below 0.00001 are converted to 0.00001.

IEDirectShow

DECLARATION

```
procedure DVDPlayAt(Title: integer; Chapter: integer);
procedure DVDPlayAt(Title: integer; Hours, Minutes,Seconds,Frames: integer);
```

DESCRIPTION

DVDPlayAt starts playback from the beginning of the specified chapter of the specified title. You can know the number of Chapters and Titles (and the current position) calling DVDGetProperty method.

The second overload allows specifying the exact time inside the specified title.

DECLARATION

```
procedure DVDSelectAt(x,y:integer);
```

DESCRIPTION

Select a button in DVD menu, at specified position. To actually press the button, call DVDActivateButton. In order to play DVD inside Delphi IDE please disable “Integrated Debugging”.

IEDirectShow

DECLARATION

```
procedure DVDShowMenu(Menu: TIEDVDMenu);
```

DESCRIPTION

DVDShowMenu displays the specified menu, if available.

DECLARATION

```
property EnableSampleGrabber: Boolean;
```

DESCRIPTION

If EnableSampleGrabber is true, then an event occurs whenever a new frame is available. The event is OnDShowNewFrame (unless you use SetNotifyWindow and use TIEDirectShow as standalone object).

DECLARATION

```
property EndOfStream: Boolean;
```

DESCRIPTION

EndOfStream tests whether the file position is at the end of a file.

DECLARATION

```
property FileInput: AnsiString;
```

DESCRIPTION

If you don't want get video from a capture card, but just from a file, you have to set FileInput property with a full path name and a multimedia file.

DECLARATION

```
property FileOutput: AnsiString;
```

DESCRIPTION

FileOutput specifies the full path of the output multimedia file.

DECLARATION

```
function GetAverageTimePerFrame: int64;
```

DESCRIPTION

GetAverageTimePerFrame returns, when available, the number of milliseconds for each frame.

DECLARATION

```
procedure GetCurrentVideoFormat(var width,
height: integer; var format: AnsiString);
```

DESCRIPTION

GetCurrentVideoFormat allows setting the video format. You can know supported formats reading VideoFormats property. The format parameter is the same of VideoFormats[].Format, or an empty string if it doesn't matter.

DECLARATION

```
function GetEventCode(var Event:
integer):Boolean;
```

DESCRIPTION

GetEventCode reads the current event obtained from DirectShow.

An OnDShowEvent occurs whenever there is an event available.
Common events are:
IEEC_COMPLETE
IEEC_USERABORT
IEEC_ERRORABORT
IEEC_TIME
IEEC_REPAINT
IEEC_STREAM_ERROR_STOPPED

IEDirectShow

IEEC_STREAM_ERROR_STILLPLAYING
IEEC_ERROR_STILLPLAYINGIEEC_PALETTE_CHANGED
IEEC_VIDEO_SIZE_CHANGED
IEEC_QUALITY_CHANGE
IEEC_SHUTTING_DOWN
IEEC_CLOCK_CHANGED
IEEC_PAUSED
IEEC_OPENING_FILE
IEEC_BUFFERING_DATA
IEEC_FULLSCREEN_LOST
IEEC_ACTIVATE
IEEC_NEED_RESTART
IEEC_WINDOW_DESTROYED
IEEC_DISPLAY_CHANGED
IEEC_STARVATION
IEEC_OLE_EVENT
IEEC_NOTIFY_WINDOW
IEEC_STREAM_CONTROL_STOPPED
IEEC_STREAM_CONTROL_STARTED
IEEC_END_OF_SEGMENT
IEEC_SEGMENT_STARTED
IEEC_LENGTH_CHANGED
IEEC_DEVICE_LOST
IEEC_STEP_COMPLETE
IEEC_SKIP_FRAMES
IEEC_TIMECODE_AVAILABLE
IEEC_EXTDEVICE_MODE_CHANGE
IEEC_GRAPH_CHANGED
IEEC_CLOCK_UNSET

IEDirectShow

DECLARATION

```
procedure GetSample(DestBitmap: TIEBitmap;
resample: Boolean = true);
```

DESCRIPTION

GetSample copies current sample to the specified bitmap.
You can call this method inside a OnDShowNewFrame event or anywhere you want (for example, on response to a TTimer event).

If resample is true (default) and input is VMR then the image is resampled to the original bitmap size.

DECLARATION

```
function GetSupportedTVStandards: TIETVStandards;
```

DESCRIPTION

Returns a set of supported tv standards (NTSC, Pal,...).
This function is useful to know supported standard before call SetTVStandard.

IEDirectShow

DECLARATION

```
procedure GetVideoRenderNativeSize(var Width: integer; var Height: integer);
```

DESCRIPTION

GetVideoRenderNativeSize returns the video frame size when RenderVideo is true. You should set the background bitmap to the same size.

DECLARATION

```
procedure GetVideoRenderRect(var SrcRect: TRect; var DstRect: TRect);
```

DESCRIPTION

Used internally by TImageEnView in order to get source and destination rectangle when RenderVideo is true.

DECLARATION

```
property NotifyWindow: THandle;
```

DESCRIPTION

Returns current notification window handle.

IEDirectShow

DECLARATION

```
procedure Pause;
```

DESCRIPTION

Pause is used to suspend the graph.

DECLARATION

```
property Position: int64;
```

DESCRIPTION

Position returns the stream (multimedia file) position in a format specified by TimeFormat property. The position is measured in 100-nanosecond per unit (when time is selected).

DECLARATION

```
property Rate: double;
```

DESCRIPTION

Specifies the frame rate where Rate is:
1 = normal speed
2 = double speed
0.5 = half speed
Not all devices support this property. **Important**. The rate does NOT set or get the number of frames per second (FPS). To save a movie originally shot at 30 fps to.15 fps set the rate to 0.5. The Rate can only be set if the if the file is running or is paused..

DECLARATION

```
property ReferenceClock: TIEReferenceClock;
```

DESCRIPTION

ReferenceClock specifies the source of reference clock. You should change it when audio and video aren't synchronized. When RenderAudio and RenderVideo are true, it is suggested to set ReferenceClock = rcAudioOutput.

DECLARATION

```
property RenderAudio: Boolean
```

DESCRIPTION

If RenderAudio is true renders the audio stream to the default audio renderer. Sometimes audio is not synchronized with video. To get audio/video synchronized we should interact directly with hardware devices, but these will loss compatibility. You could try to play with ReferenceClock and ClockErrorTolerance properties to get better synchronization.

DECLARATION

```
property RenderVideo: Boolean;
```

DESCRIPTION

When true the video input (from capture card, file or dvd) is displayed inside the component area. This functionality is available only starting from Windows XP.

Demos

capture\VMR_camera
capture\VMR_capture
capture\VMR_DVD
capture\VMR_video

DECLARATION

```
procedure RepaintVideo(hwnd: THandle; hdc:
THandle);
```

DESCRIPTION

Used internally by TImageEnView in order to repaint display when RenderVideo is true.

DECLARATION

```
procedure Run;
```

DESCRIPTION

After calling Connect you have to run the graph. This starts video capture or file capture.

DECLARATION

```
procedure SaveGraph(filename: AnsiString);
```

DESCRIPTION

This method is useful to debug the filter graph built using Connect. The file must have 'grf' extension and can be read using GraphEdit utility (you can get this from the Microsoft DirectShow SDK).

DECLARATION

```
procedure SetAudioCodec(idx: integer);
```

DESCRIPTION

SetAudioCodec specifies the audio compression codec. You can get an index from AudioCodecs property.

DECLARATION

```
procedure SetAudioInput(idx: integer;
instanceIndex: integer=0);
```

DESCRIPTION

SetAudioInput allows selection of an audio input. You can get an index from AudioInputs property. InstanceIndex specifies the audio input on the same device.

DECLARATION

```
procedure SetCurrentVideoFormat(width, height:
integer; format: AnsiString = '');
```

DESCRIPTION

SetCurrentVideoFormat allows setting the video format. You can know supported formats reading VideoFormats property. The format parameter is the same of VideoFormats.Format, or an empty string if it doesn't matter.

Please, to set frame resolution on webcams use additional parameters of SetVideoInput instead of SetCurrentVideoFormat.

DECLARATION

```
procedure SetNotifyWindow(WindowHandle: THandle;
NewFrameMessage: integer; EventMessage: integer);
```

DESCRIPTION

For internal use only. Used only if you use TIEDirectShow as standalone object.

DECLARATION

```
procedure SetTVStandard(Value: TIETVStandard);
```

DESCRIPTION

Sets the input tv standard. To know supported tv standards use GetSupportedTVStandards.

DECLARATION

```
procedure SetVCRHorizontalLocking(Value:
Boolean);
```

DESCRIPTION

When false assumes input from broadcast source (better synchronization pulses), when true tells the decoder to relax its standards (typically used from a tape source).

DECLARATION

```
procedure SetVideoCodec(idx: integer);
```

DESCRIPTION

SetVideoCodec specifies the video compression codec. You can get an index from VideoCodecs property.

DECLARATION

```
procedure SetVideoInput(idx :integer;
instanceIndex: integer = 0; width: integer = 0;
height: integer = 0; format: AnsiString = '');
```

DESCRIPTION

SetVideoInput allows selection of a video input. You can get an index from VideoInputs property. InstanceIndex specifies the video input on the same device. Width and height specifies video frame size in pixels. Format specifies video format (ie. 'YUY2',...)

DECLARATION

```
procedure SetVideoRenderRect(SrcRect, DstRect:
TRect);
```

DESCRIPTION

Used internally by TImageEnView in order to set source and destination rectangle when RenderVideo is true.

DECLARATION

```
function ShowPropertyPages(proppages:
TIEPropertyPages; proptype: TIEPropertyPagesType;
checkOnly: Boolean=false): Boolean;
```

DESCRIPTION

ShowPropertyPages shows a dialog for the specified filter of the filter graph. Proppages specifies the filter, while proptype specifies if you want the dialog for the input, output or common filter properties. If checkOnly is true, the dialog will not be displayed. It is useful to know if the dialog is available (checked the return value). ShowPropertyPages return false if fails.

DECLARATION

```
function State: TIEDirectShowState;
```

DESCRIPTION

State function returns the current state of the graph.

DECLARATION

```
procedure Step(frames: integer);
```

DESCRIPTION

Go forward for frames steps.

DECLARATION

```
procedure Stop;
```

DESCRIPTION

This is used to stop the graph's execution.

DECLARATION

```
property TimeFormat: TIETimeFormat;
```

DESCRIPTION

TimeFormat specifies the time format for Position and Duration properties.

IEDirectShow

DECLARATION

```
property TunerChannel: integer;
```

DESCRIPTION

TunerChannel sets the channel when the video input source is a TV Tuner.

DECLARATION

```
function TunerFindSignal: Boolean;
```

DESCRIPTION

Returns True when a channel contains a TV signal.

DECLARATION

```
property VideoCodecQuality: double;
```

DESCRIPTION

VideoCodecQuality specifies the video compression codec quality. Not all codecs support this property.

DECLARATION

```
property VideoCodecs: TStringList;
```

DESCRIPTION

VideoCodecs contains a list of video compression codecs.

DECLARATION

```
property VideoFormats[i: integer]:
TIEVideoFormat;
```

DESCRIPTION

VideoFormats contains a list of available video formats.
A video format specifies the frame width, height and other info.

DECLARATION

```
property VideoFormatsCount: integer;
```

DESCRIPTION

VideoFormatsCount specifies the VideoFormats list size.

DECLARATION

```
property VideoInputs: TStringList;
```

DESCRIPTION

Contains a list of available video inputs (video capture inputs).

DECLARATION

```
property VideoInputSource: integer;
```

DESCRIPTION

Specifies the video input source. A video input source is a line input, like Video-Composite, Tuner, etc. You can get an index from VideoInputSources property.

DECLARATION

```
property VideoInputSources: TStringList;
```

DESCRIPTION

VideoInputSources contains a list of video input sources available for the selected video input. A video input source is a line input, like Video-Composite, Tuner, etc.

IEDirectShow

Chapter 21

TIEMediaReader

Chapter 21. IEMediaReader

The TIEMediaReader class allows to get frames from a media file (wmv, mpeg...). TIEMediaReader is a simple stand-alone class, with one single method to copy the wanted frame to a TIEBitmap object. Reading of frames could be slow.

Example

```
var
media: TIEMediaReader;
begin
  media  := TIEMediaReader.Create('video.wmv');
  try
    // get frame 0 and save to sample0.jpeg
    media.GetSample( 0, ImageEnView1.IEBitmap );
    ImageEnView1.IO.SaveToFile('sample0.jpeg');
    // get frame 1 and save to sample1.jpeg
    media.GetSample( 1, ImageEnView1.IEBitmap );
```

```
  ImageEnView1.IO.SaveToFile('sample1.jpeg');
  //...etc...
  finally
    media.free;
  end;
end;
```

Methods and Properties

FrameCount	FrameHeight	FrameRate
FrameWidth	GetSample	Length

Methods and Properties

DECLARATION

```
property ConnectedDeviceIndex: integer;
```

DESCRIPTION

ConnectedDeviceIndex returns the currently connected device index. This property is useful when you connect to a device using Windows dialog, to know which device has been connected, so next time it can be connected directly.

Unit IEDS

TIEMediaReader

DECLARATION

```
function ConnectTo(idx: integer):Boolean;
```

DESCRIPTION

Connect to the specified device. Returns True if successful.

DECLARATION

```
function ConnectToUsingDialog: Boolean;
```

DESCRIPTION

ConnectToUsingDialog shows a system dialog that allows users to select a WIA device. Returns true if user presses OK.

DECLARATION

```
procedure DeleteItem(item: TIEWiaItem);
```

DESCRIPTION

DeleteItem deletes the specified item. At the end of session the item should be actually removed.

Demo
capture\camerageti mages

IEMediaReader

DECLARATION

```
property DeleteTakenPicture: Boolean;
```

DESCRIPTION

If true, the taken picture (using TakePicture> will be not saved in the camera memory.

Demo
capture\camerageimages

DECLARATION

```
property Device: TIEWiaItem;
```

DESCRIPTION

Device returns the currently connected (selected) device. This is the 'root' item.

DECLARATION

```
property DevicesInfo[idx: integer]:
TIEWiaDeviceInfo;
```

DESCRIPTION

DevicesInfo returns info about the specified WIA device.
TIEWiaDeviceInfo is defined with:

Unit IEDS
TIEMediaReader

DECLARATION

```
property DevicesInfoCount: integer;
```

DESCRIPTION

DevicesInfoCount returns the number of WIA devices available.

DECLARATION

```
procedure FillTreeView(Items: TTreeNodes;
IncludeDescription: Boolean);
```

DESCRIPTION

FillTreeView fills a TTreeView component with the content of WIA device.

DECLARATION

```
function GetDeviceProperty(PropId: TPropID):
Variant;
```

DESCRIPTION

GetDeviceProperty gets a device property value as Variant.

IEMediaReader

Unit IEDS
TIEMediaReader

DECLARATION

```
function GetItemProperty( PropId: TPropID; item:
TIEWiaItem=nil ): Variant;
```

DESCRIPTION

GetItemProperty gets an item (image) property as Variant.
WIA item properties

DECLARATION

```
procedure GetItemPropertyAttrib( PropId: TPropID;
item: TIEWiaItem; var attrib: TIEWiaAttrib; var
values: TIEWiaValues );
```

DESCRIPTION

Fills attrib and values parameters. They allow knowing a property range and access rights. WIA item properties

IEMediaReader

Declaration

```
function GetItemThumbnail(item: TIEWiaItem;
destBitmap: TIEBitmap):Boolean;
```

Description

GetItemThumbnail downloads the thumbnail of specified item in destBitmap. GetItemThumbnail returns true on success.

Demo
capture\camerageimages

Declaration

```
function IsItemDeleted(item: TIEWiaItem):
Boolean;
```

Description

IsItemDeleted returns true when the item is marked as deleted.

Demo
capture\camerageimages

DECLARATION

```
property SaveTransferBufferAs: string
```

DESCRIPTION

SaveTransferBufferAs specifies a filename where to transfer the raw data received from the WIA device. This is useful when you want jpegs from a Camera, without lose data.

Demo
capture\camerageimages

DECLARATION

```
function SetDeviceProperty(PropId: dword; val: integer): Boolean;
```

DESCRIPTION

SetDeviceProperty sets a device property value as integer.
WIA device properties.

Unit IEDS

TIEMediaReader

Declaration

```
function SetDevicePropertyVariant(PropId: dword;
val: Variant): Boolean;
```

Description

SetDevicePropertyVariant sets a device property value as Variant. WIA device properties.

Declaration

```
function TIEWia.SetItemProperty(PropId: dword;
val: integer; item: TIEWiaItem=nil): Boolean;
```

Description

SetItemProperty sets an item (image) property as integer. WIA item properties

IEMediaReader

Declaration

```
function SetItemPropertyVariant(PropId: dword;
val: Variant; item: TIEWiaItem=nil): Boolean;
```

Description

SetItemPropertyVariant sets an item (image) property as Variant. WIA item properties

Declaration

```
function ShowAcquireDialog(SystemDialog:
Boolean):Boolean;
```

Description

Shows a system dialog that allows user to set parameters and display a preview of the image. You still need to call Acquire to get the image. Returns true if user presses OK (or Acquire).

DECLARATION

```
property TakePicture: Boolean;
```

DESCRIPTION

If TakePicture is true, the next acquire will take a new picture from the camera.

Demo
capture\camerageimages

DECLARATION

```
property TransferFormat: TIETransferFormat;
```

DESCRIPTION

TransferFormat specifies a filename where to transfer the raw data received from the WIA device. This is useful when you want jpegs from a Camera, without lose data or a BMP from a scanner for maximum quality.

Demo
capture\camerageimages

DECLARATION

```
procedure UpdateItems;
```

DESCRIPTION

Reload items tree.

Unit IEDS
TIEMediaReader

DESCRIPTION

This class allows getting frames from a media file (wmv, mpeg...). TIEMediaReader is a simple stand-alone class, with one single method to copy the wanted frame to a TIEBitmap object.
Reading of frames could be slow.

Methods

DECLARATION

```
property FrameCount: int64;
```

DESCRIPTION

Returns the number of frames (samples).

DECLARATION

```
property FrameHeight: integer;
```

DESCRIPTION

Returns the frame bitmap height.

DECLARATION

```
property FrameRate: double;
```

DESCRIPTION

Returns the frame rate.

DECLARATION

```
property FrameWidth: integer;
```

DESCRIPTION

FrameWidth returns the frame bitmap width.

DECLARATION

```
procedure GetSample(frame: int64; OutBitmap:
TIEBitmap);
```

DESCRIPTION

Copies the sample indexed by frame to OutBitmap.

Unit IEDS
TIEMediaReader

DECLARATION

```
property Length: double;
```

DESCRIPTION

Returns the video length;

Chapter 22

TIETwainParams

Chapter 22. IETwainParams

Description

The TIETWainParams object contains many properties and methods to control TWain scanners without using the default dialog.

Unit ImageENIO

TIETwainParams

Methods and Properties

TWain Communication Handling

Assign	Create	FreeResources
GetDefaultSource	GetFromScanner	SelectSourceByName
SetDefaultParams	Update	

TWain Properties

AcquireFrameBottom	AcquireFrameEnabled	AcquireFrameLeft
AcquireFrameRight	AcquireFrameTop	AppManufacturer
AppProductFamily	AppProductName	AppVersionInfo
AutoBorderDetection	AutoBright	AutoDeskew
AutoDiscardBlankPages	AutoFeed	AutoRotate
BitDepth	Brightness	BufferedTransfer
CapabilitiesValid	CompatibilityMode	Contrast
Country	DuplexEnabled	DuplexSupported
FeederEnabled	FeederLoaded	FileTransfer
Filter	Gamma	Highlight
Language	LastErrorStr	LastError

TWain Properties

LogFile	Orientation	PaperDetectable
PhysicalHeight	PhysicalWidth	PixelType
ProgressIndicators	Rotation	SelectedSource
Shadow	ShowSettingsOnly	SourceCount
SourceName	StandardSize	Threshold
UndefinedImageSize	UseMemoryHandle	VisibleDialog
XResolution	XScaling	YResolution
YScaling		

Twain Communication Handling

DECLARATION

```
procedure Assign(Source: TIETWainParams);
```

DESCRIPTION

Applications can Assign TWain parameters between TImageEnIO and TImageEnMIO components.

IETWainParams

DECLARATION

```
constructor Create(Owner: TComponent);
```

DECLARATION

```
procedure FreeResources;
```

DESCRIPTION

ImageEn keeps the scanner driver open to improve scanning performance. Call FreeResources to close and free the scanner driver after a call to acquire method: this degrades performance but can improve stability. Applications should not call FreeResources because it is executed when the application exits.

DECLARATION

```
function GetDefaultSource: integer;
```

DESCRIPTION

GetDefaultSource returns the system's default twain source. The default ImageEn source has the index 0, but this should be different from the system's default source.

DECLARATION

```
function GetFromScanner: Boolean;
```

DESCRIPTION

GetFromScanner fills all properties with scanner parameters. Use this method to update ImageEn parameters with scanner values. Returns False if cannot connect to the scanner.

DECLARATION

```
function SelectSourceByName(const sn:
AnsiString): Boolean;
```

DESCRIPTION

SelectSourceByName selects the first device that matches left name string with sn. Returns true if device is found. The list of names is contained in SourceName[] list.

DECLARATION

```
procedure SetDefaultParams;
```

DESCRIPTION

SetDefaultParams restores default parameters.

IETWainParams

DECLARATION

```
procedure Update;
```

DESCRIPTION

The Update method determines if current parameters are valid.

For example, if the application assigns a combination of PixelType and YResolution unsupported by scanner, the Update method restores the PixelType and YResolution value to a value supported by the scanner.

Twain Properties

DECLARATION

```
property AcquireFrameBottom:  double;
```

DESCRIPTION

AcquireFrameBottom is the bottom of the rectangle to acquire measured in inches.

See also
AcquireFrameLeft
AcquireFrameTop
AcquireFrameRight

DECLARATION

```
property AcquireFrameEnabled: Boolean;
```

DESCRIPTION

If AcquireFrameEnabled is True, it enables the properties AcquireFrameLeft, AcquireFrameRight, AcquireFrameTop and AcquireFrameBottom.

When true, some scanners don't allow the user to change the acquisition frame. Default is False.

DECLARATION

```
property AcquireFrameLeft: double;
```

DESCRIPTION

AcquireFrameLeft is the left of the rectangle to acquire measured in inches.

See also
AcquireFrameTop
AcquireFrameRight
AcquireFrameBottom

DECLARATION

```
property AcquireFrameRight: double;
```

DESCRIPTION

AcquireFrameRight is the right side of the rectangle to acquire measured in inches.

See also
AcquireFrameLeft
AcquireFrameTop
AcquireFrameBottom

DECLARATION

```
property AcquireFrameTop: double;
```

DESCRIPTION

AcquireFrameTop is the top of the rectangle to acquire measured in inches.

See also
AcquireFrameLeft
AcquireFrameRight
AcquireFrameBottom

DECLARATION

```
property AppManufacturer: AnsiString;
```

DESCRIPTION

These properties specify the application information communicated to the Twain scanner interface.

DECLARATION

```
property AppProductfamily: AnsiString;
```

DESCRIPTION

These properties specify the application information communicated to the Twain scanner interface.

DECLARATION

```
property AppProductName: AnsiString;
```

DESCRIPTION

These properties specify the application information communicated to the Twain scanner interface.

IETWainParams

DECLARATION

```
property AppVersionInfo: AnsiString;
```

DESCRIPTION

These properties specify the application information communicated to the Twain scanner interface.

DECLARATION

```
property AutoBorderDetection: Boolean;
```

DESCRIPTION

AutoBorderDetection turns automatic border detection on and off.

DECLARATION

```
property AutoBright: Boolean;
```

DESCRIPTION

TRUE enables and FALSE disables the Source's Auto-brightness function (if any).

Unit ImageENIO
TIETwainParams

DECLARATION

```
property AutoDeskew: Boolean;
```

DESCRIPTION

AutoDeskew turns automatic deskew correction on and off.

Unit ImageENIO
TIETwainParams

DECLARATION

```
property AutoDiscardBlankPages: integer;
```

DESCRIPTION

Use this capability to have the scanner discard blank images. Applications never see these images during the scanning session. Allowed values:

Value	Description
-2	Indicates that all images will be delivered to the application, none of them will be discarded
-1	Scanner source will decide if an image is blank or not and discard as appropriate
>=0	Scanner will use it as the byte size cutoff point to identify which images are to be discarded. If the size of the image is less than or equal to this value, then it will be discarded. If the size of the image is greater than this value, then it will be kept so that it can be transferred to the Application.

DECLARATION

```
property AutoFeed: Boolean;
```

DESCRIPTION

If AutoFeed is true, the scanner will automatically feed the next page from the document feeder.

DECLARATION

```
property AutoRotate: Boolean;
```

DESCRIPTION

When AutoRotate is true this capability depends on intelligent features within the Source to automatically rotate the image to the correct position.

DECLARATION

```
property BitDepth: TIEIntegerList;
```

DESCRIPTION

BitDepth specifies the bit depth (bits per channel) of the image to scan.

Unit ImageENIO
TIETwainParams

DECLARATION

```
property Brightness: TIEDoubleList;
```

DESCRIPTION

Brightness value. Allowed range is -1000 to +1000.

Allowed values can be assigned to Brightness.CurrentValue property. To see which values your scanner supports look at Brightness.Items[] array, or Brightness.RangeMin, Brightness.RangeMax and Brightness. RangeStep. You can also limit scanner user interface allowed values removing some Brightness.Items[] items or changing Brightness.RangeMin, Brightness.RangeMax and Brightness.RangeStep.

DECLARATION

```
property BufferedTransfer: Boolean;
```

DESCRIPTION

Set the BufferedTransfer property to False if you have problems acquiring an image. You never use this property under normal circumstances.

IETWainParams

DECLARATION

```
property CapabilitiesValid: Boolean;
```

DESCRIPTION

If true all properties are updated from scanner, otherwise not and you should call GetFromScanner to update them.

DECLARATION

```
property CompatibilityMode: Boolean;
```

DESCRIPTION

CompatibilityMode disables capability setting and reading (some scanner may crash when you set/get capabilities).
Set this property to true only if you have problems with a specific scanner.

DECLARATION

```
property Contrast:  TIEDoubleList;
```

DESCRIPTION

Contrast value. Allowed range is -1000 to +1000.

Allowed values can be assigned to Contrast.CurrentValue property. To see which values your scanner supports look at the Contrast.Items[] array, or Contrast.RangeMin, Contrast.RangeMax and Contrast.RangeStep. You can also limit scanner user interface allowed values by removing some Contrast.Items[] items or changing Contrast.RangeMin, Contrast.RangeMax or Contrast.RangeStep.

DECLARATION

```
property Country: word;
```

DESCRIPTION

Specifies the primary country where your application is intended to be distributed. Default is TWLG_USA.
Look also TWain country constants.

DECLARATION

```
property DuplexEnabled: Boolean;
```

DESCRIPTION

If DuplexEnabled is true, the scanner scans both sides of a paper; otherwise (default), the scanner will scan only one side.
Use this property only within TImageEnMIO component to enable/disable duplex mode.

DECLARATION

```
property DuplexSupported: Boolean;
```

DESCRIPTION

If DuplexSupported is true, the scanner can scans both sides of a paper; otherwise the scanner will scan only one side.

DECLARATION

```
property FeederEnabled: Boolean;
```

DESCRIPTION

FeederEnabled enables the feed loader mechanism when present. Use this property only within TImageEnMIO component to disable the feed loader.

DECLARATION

```
property FeederLoaded: Boolean;
```

DESCRIPTION

Use the FeederLoaded property to reflect whether or not there are documents loaded in the Source's feeder.

Unit ImageENIO
TIETwainParams

DECLARATION

```
property FileTransfer: Boolean;
```

DESCRIPTION

If FileTransfer true, uses a file transfer to get images from the scanner (regardless of BufferedTransfer setting).
There are three ways to get an image from scanner:
1 - native transfer (set FileTransfer = False and BufferedTransfer=False)
2 - buffered transfer (set FileTransfer = False and BufferedTransfer = True)
3 - file transfer (set FileTransfer=True)

FileTransfer is slow but more compatible.

DECLARATION

```
property Filter: TIETWFilter;
```

DESCRIPTION

Filter describes the color characteristic of the subtractive filter applied to the image data. IetwUndefined (default) unsets this property, while ietwNone set “none” value.

DECLARATION

```
property Gamma:  double;
```

DESCRIPTION

Gamma is the gamma value of your scanner.

Read-only

DECLARATION

```
property Highlight: double;
```

DESCRIPTION

Specifies which value in an image should be interpreted as the lightest “highlight”. All values “lighter” than this value will be clipped to this value. -1 is the default scanner value (it means “do not change current value”). Allowed values from 0 to 255.

DECLARATION

```
property Language: word;
```

DESCRIPTION

Specifies the primary language for the scanner dialogs. Default is TWLG_USERLOCALE. Look also TWain language constants.
Versions before 3.0.3 have TWLG_USA for default.

DECLARATION

```
LastErrorStr:  AnsiString;
```

DESCRIPTION

LastErrorStr returns the last error which occurred on scanning as a string (see also LastError).

Unit ImageENIO

TIETwainParams

DECLARATION

```
LastError: integer;
```

DESCRIPTION

LastError returns the last error which occurred on scanning. Allowed values (defined in ietwain unit) are:

```
TWCC_SUCCESS = 0; // It worked!
TWCC_BUMMER = 1; // Failure due to unknown causes
TWCC_LOWMEMORY = 2; // Not enough memory to perform
operation
TWCC_NODS = 3; // No Data Source
TWCC_MAXCONNECTIONS = 4; // DS is connected to max
possible applications
TWCC_OPERATIONERROR = 5; // DS or DSM reported
error, application shouldn't
TWCC_BADCAP = 6; // Unknown capability
TWCC_BADPROTOCOL = 9; // Unrecognized MSG DG DAT
combination
TWCC_BADVALUE = 10; // Data parameter out of range
TWCC_SEQERROR = 11; // DG DAT MSG out of expected
sequence
TWCC_BADDEST = 12; // Unknown destination
Application/Source in DSM_Entry
TWCC_CAPUNSUPPORTED = 13; // Capability not
supported by source
TWCC_CAPBADOPERATION = 14; // Operation not
supported by capability
TWCC_CAPSEQERROR = 15; // Capability has dependency
on other capability
TWCC_DENIED = 16; // File System operation is
denied (file is protected)
```

```
TWCC_FILEEXISTS = 17; // Operation failed because
file already exists.
```
TWCC_FILENOTFOUND = 18; // File not found
TWCC_NOTEMPTY = 19; // Operation failed because directory is not empty
TWCC_PAPERJAM = 20; // The feeder is jammed
TWCC_PAPERDOUBLEFEED = 21; // The feeder detected multiple pages
TWCC_FILEWRITEERROR = 22; // Error writing the file (meant for things like disk full conditions)
TWCC_CHECKDEVICEONLINE = 23; // The device went offline prior to or during this operation

DECLARATION

```
property LogFile: string;
```

DESCRIPTION

LogFile specifies the file name that will contain the log of the communication between ImageEn and the scanner driver.
Set this property before use scanner related methods or properties.

Unit ImageENIO
TIETwainParams

DECLARATION

```
property Orientation: TIEIntegerList;
```

DESCRIPTION

Orientation defines the orientation of the output image. Not all scanners support this capability.
Valid values are:

Value	Description
0	No rotation (Portrait)
1	Rotate 90°
2	Rotate 180°
3	Rotate 270° (Landscape)

The above values can be assigned to the Orientation.CurrentValue property. To know which values your scanner supports, look at Orientation.Items[] array.

DECLARATION

```
property PaperDetectable: Boolean;
```

DESCRIPTION

If PaperDetectable is true, the scanner is able to detect paper.

DECLARATION

```
property PhysicalHeight: double;
```

DESCRIPTION

PhysicalHeight is the maximum physical height (Y-axis) the scanner can acquire (measured in inches).

Read-only

DECLARATION

```
property PhysicalWidth: double;
```

DESCRIPTION

PhysicalWidth is the maximum physical width (X-axis) the scanner can acquire (measured in inches).

Read-only

Unit ImageENIO
TIETwainParams

DECLARATION

```
property PixelType: TIEIntegerList;
```

DESCRIPTION

PixelType is the type of pixel data that a scanner is capable of acquiring. Valid values are:

Value	Description
0	Black & white (1 bit)
1	Gray scale (8 bit)
2	Full RGB (24 bit)
3	Palette (ImageEn can't support this)
4	CMY (ImageEn can't support this)
5	CMYK (ImageEn can't support this)
6	YUV (ImageEn can't support this)
7	YUVK (ImageEn can't support this)
8	CIEXYZ (ImageEn can't support this)

Above values can be assigned to PixelType.CurrentValue property. To see which values your scanner supports, look at PixelType.Items[] array. You can also limit scanner user interface allowed values by removing some PixelTtype.Items[] items.

IETWainParams

Unit ImageENIO
TIETwainParams

DECLARATION

```
property ProgressIndicators:  Boolean;
```

DESCRIPTION

If ProgressIndicators is True (default), the scanner will display a progress indicator during acquisition and transfer, regardless of whether the scanner user interface is active. You can use OnProgress event to display your custom progress indicator.

DECLARATION

```
property Rotation: TIEDoubleList;
```

DESCRIPTION

Specifies how much the Source can/should rotate the scanned image data prior to transfer. Allowed values are in the range: -360..+360 degrees.

DECLARATION

```
property SelectedSource: integer;
```

DESCRIPTION

SelectedSource is an index of SourceName[] list, and defines the currently selected scanner (source).

The default is 0 (first scanner). This is an alternative method to select a scanner without calling the SelectAcquireSource method.

DECLARATION

```
property Shadow: double;
```

DESCRIPTION

Specifies which value in an image should be interpreted as the darkest “shadow”. All values “darker” than this value will be clipped to this value. -1 is the default scanner value (it means “do not change current value”). Allowed values from 0 to 255.

DECLARATION

```
property ShowSettingsOnly: Boolean;
```

DESCRIPTION

If ShowSettingsOnly is True the scanner driver will show settings dialog, without acquire the image. You could use SourceSettings to read scanner settings.

Demo
capture\twain store

DECLARATION

```
property SourceCount: integer;
```

DESCRIPTION

SourceCount is the count of the installed scanners. It also defines the length of the SourceName[] list.

Read-only

DECLARATION

```
property SourceName[index]: AnsiString;
```

DESCRIPTION

SourceName is the name of the index scanner. This list has SourceCount values.

Read-only

DECLARATION

```
property StandardSize: TIEIntegerList;
```

DESCRIPTION

StandardSize page size for devices that support fixed frame sizes. Defined sizes match typical page sizes. This specifies the size(s) the Source can/should use to acquire image data. Allowed values:

```
IETW_NONE
IETW_A4LETTER
IETW_B5LETTER
IETW_USLETTER
IETW_USLEGAL
IETW_A5
IETW_B4
IETW_B6
IETW_USLEDGER
IETW_USEXECUTIVEIETW_A3
```

IETWainParams

```
IETW_B3
IETW_A6IETW_C4
IETW_C5
IETW_C6
IETW_4A0
IETW_2A0
IETW_A0
IETW_A1
IETW_A2
IETW_A4
IETW_A7
IETW_A8
IETW_A9
IETW_A10
IETW_ISOB0
IETW_ISOB1
IETW_ISOB2
IETW_ISOB3
IETW_ISOB4
IETW_ISOB5
IETW_ISOB6
IETW_ISOB7
IETW_ISOB8
IETW_ISOB9
IETW_ISOB10
IETW_JISB0
IETW_JISB1
IETW_JISB2
IETW_JISB3
IETW_JISB4
IETW_JISB5
IETW_JISB6
IETW_JISB7
```

IETWainParams

```
IETW_JISB8
IETW_JISB9
IETW_JISB10
IETW_C0
IETW_C1
IETW_C2
IETW_C3
IETW_C7
IETW_C8
IETW_C9
IETW_C10
IETW_USSTATEMENT
IETW_BUSINESSCARD
```

DECLARATION

```
property Threshold: TIEDoubleList;
```

DESCRIPTION

Threshold specifies the dividing line between black and white. Allowed values: 0..255.

DECLARATION

```
property UndefinedImageSize: Boolean;
```

DESCRIPTION

UndefinedImageSize enables support for an undefined image size scanner. Default is False.

DECLARATION

```
property UseMemoryHandle: Boolean;
```

DESCRIPTION

Some scanner's drivers do not support memory handles, so we have to pass memory pointers in order to transfer images.
You should try to set this property to False if you have problems scanning documents or images.

DECLARATION

```
property VisibleDialog: Boolean;
```

DESCRIPTION

If VisibleDialog is True (default), the scanner user interface is enabled when Acquire method is called.

DECLARATION

```
property XResolution: TIEDoubleList;
```

DESCRIPTION

XResolution is the DPI (Dots per Inch) in the X-axis.

Allowed values can be assigned to XResolution.CurrentValue property. To see which values your scanner supports, look at the XResolution.Items[] array, or XResolution.RangeMin, XResolution.RangeMax and XResolution.RangeStep.

You can also limit scanner user interface allowed values by removing some XResolution.Items[] items or changing XResolution.RangeMin, XResolution.RangeMax and XResolution.RangeStep.

DECLARATION

```
property XScaling: TIEDoubleList;
```

DESCRIPTION

XScaling is the X-axis scaling value. A value of 1.0 is equivalent to 100% scaling.

Allowed values can be assigned to XScaling.CurrentValue property. To see which values your scanner supports, look at XScaling.Items[] array, or XScaling.RangeMin, XScaling.RangeMax and XScaling.RangeStep.

You can also limit scanner user interface allowed values by removing some XResolution.Items[] items or by changing XResolution.RangeMin, XResolution.RangeMax and XResolution.RangeStep.

IETWainParams

Unit ImageENIO
TIETwainParams

DECLARATION

```
property YResolution: TIEDoubleList;
```

DESCRIPTION

YResolution is the DPI (Dots per Inch) in the Y-axis.

Allowed values can be assigned to YResolution.CurrentValue property. To see which values your scanner supports, look at YResolution.Items[] array, or YResolution.RangeMin, YResolution.RangeMax and YResolution.RangeStep.

You can also limit scanner user interface allowed values by removing some YResolution.Items[] items or by changing YResolution.RangeMin, YResolution.RangeMax or YResolution.RangeStep.

Unit ImageENIO
TIETwainParams

DECLARATION

```
property YScaling: TIEDoubleList;
```

DESCRIPTION

YScaling is the Y-axis scaling value. A value of 1.0 is equivalent to 100% scaling.

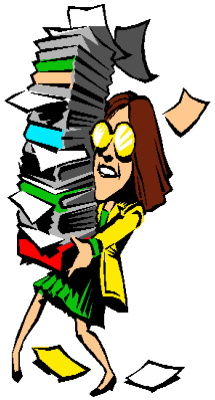
Allowed values can be assigned to the YScaling.CurrentValue property. To see which values your scanner supports, look at the YScaling.Items[] array, or YScaling.RangeMin, YScaling.RangeMax and YScaling.RangeStep.

You can also limit scanner user interface allowed values by removing some YScaling.Items[] items or by changing YResolution.RangeMin, YResolution.RangeMax and YResolution.RangeStep.

Chapter 23

ImageEnDBView

Chapter 23. ImageEnDBView



TImageEnDBView is a descendant of TImageEnView, but it can be connected to a TDataset object to store/load images (BMP, PCX, GIF, JPEG, TIFF, PNG, PXM and others) into Blob fields or into path reference (string) fields.

TImageEnDBView works similarly to Delphi TDBImage components. You can specify particular file format parameters through IOParams property, or by executing the DoIOPreview method.

You can attach a TImageEnProc component for processing the image contained in the Blob.

See also
TImageEnView.

Database

TImageEnDBView cannot store/retrieve multiple pages in a blob; however, there is a solution. Put a TImageEnMView component on the form and use it to load/save multiple pages as TIFFs in blobs using streams. To store the content of TImageEnMView inside a blob write:

```
var
tempStream: TMemoryStream;
begin
  tempStream := TMemoryStream.Create;
  ImageEnMView1.MIO.SaveToStreamTIFF( tempStream );
  BlobField.LoadFromStream( tempStream );
  tempStream.free;
end;
```

To retrieve from a blob:

```
tempStream := TMemoryStream.Create;
BlobField.SaveToStream( tempStream );
tempStream.Position:=0;
ImageEnMView1.MIO.LoadFromStreamTIFF( tempStream );
tempStream.free;
```

If you use a TDataSet inherited data set you can create a blob stream without using intermediate TMemoryStream: this will speed up operations.

Example

```
BlobStream := myDataSet.CreateBlobStream(field, bmWrite);
ImageEnMView1.MIO.SaveToStreamTIFF( BlobStream );
BlobStream.Free;
```

```
BlobStream := myDataSet.CreateBlobStream( field,bmRead);
ImageEnMView1.MiO.LoadFromStreamTIFF( BlobStream );
BlobStream.Free;
```

Finally, when save you can set compression info. For example, for black/white images you could write:

```
ImageEnMView1.MIO.Params[0].TIFF_Compression:=
ioTIFF_G4FAX; ImageEnMView1.MIO. DuplicateCompressionInfo;
//just before save.
```

Methods and Properties

AbsolutePath	AutoDisplay	DataField
DataFieldImageFormat	DataSource	DoIOPreview
Field	IOParams	IOPreviewsParams
JpegQuality	LoadedFieldImageFormat	LoadPicture
PreviewFont	ReadOnly	StreamHeaders
OnBeforeLoadImage	OnUnableToLoadImage	

Declaration

```
property AbsolutePath: string;
```

Description

The AbsolutePath property sets/gets the base path where the stored images are when DataField points to a string field.

The final path is calculated by concatenation of AbsolutePath and string field.

The default value is an empty string. If the AbsolutePath property is empty, then the string field should be an absolute path.

Unit ImageENIO
Database

DECLARATION

```
property AutoDisplay: Boolean;
```

DESCRIPTION

AutoDisplay determines whether to automatically display the contents of a graphic BLOB in the database image control.

If AutoDisplay is True (the default value), the image automatically displays new data when the underlying BLOB field changes (such as when moving to a new record).

If AutoDisplay is False, the image clears whenever the underlying BLOB field changes.

To display the data, the user can double-click on the control or select it and press Enter. In addition, calling the LoadPicture method ensures that the control is showing data.

Change the value of AutoDisplay to False if the automatic loading of BLOB fields seems to take too long.

Declaration

```
property DataField: string;
```

Description

DataField specifies the field from which the database image displays data.

Use DataField to bind the image control to a field in the dataset. To fully specify a database field, both the dataset and the field within that dataset must be defined.

The DataSource property of the image control specifies the dataset which contains the DataField. DataField should specify a graphic field.

Declaration

```
property DataFieldImageFormat:
TDataFieldImageFormat;
```

Description

DataFieldImageFormat sets the image format to save in the Blob field or path reference.

Look at IOParams property for specific image format parameters.

Unit ImageENIO

Database

DECLARATION

```
property DataSource: TDataSource;
```

DESCRIPTION

DataSource links the image control to a dataset.

Use DataSource to link the image control to a dataset in which the data can be found.

To fully specify a database field for the image control, both the dataset and a field within that dataset must be defined. Use the DataField property to specify the particular field within the dataset.

DECLARATION

```
function DoIOPreview: Boolean;
```

DESCRIPTION

This function executes the IOPreviews dialog. This dialog gets/sets the parameters of image file formats. The dialog shown depends on the DataFieldImageFormat property.

Unit ImageENIO

Database

Declaration

```
property Field: TField;
```

Description

Field is the TField component the database image is linked to.

Read-only

Declaration

```
property IOParams: TIOParamsVals;
```

Description

IOParams allow you to set or get all file format parameters such as bits per pixel or type of compression.

Read-only

Example

```
Table1.Edit;
ImageEnDBView1.IOParams.BMP_Compression := ioBMP_RLE;
Table1.Post;
```

Unit ImageENIO

Database

DECLARATION

```
property IOPreviewsParams: TIOPreviewsParams;
```

DESCRIPTION

This property contains some features that the input/output preview dialog will have.

DECLARATION

```
property JpegQuality:  integer;
```

DESCRIPTION

JpegQuality sets the quality factor, from 1 to 100. The higher the value, the better the image quality and the larger resultant memory needed.

This is the example of IOParams.JPEG_Quality.

Declaration

```
function LoadedFieldImageFormat:
TDataFieldImageFormat;
```

Description

LoadedFieldImageFormat gets the image format stored in the Blob field (load directly from blob).

If you change DataFieldImageFormat, to store as several image formats, LoadedFieldImageFormat maintains the original value.

Declaration

```
procedure LoadPicture;
```

Description

LoadPicture loads the image stored in the field into the database image control.

DECLARATION

```
property PreviewFont: TFont;
```

DESCRIPTION

PreviewFont contains the font used in the IOPreviews dialog. Make sure the size of font matches the labels length.

Example

ImageEnDBView1.PreviewFont.Name:='MS Times New Roman';
ImageEnDBView1.DoPreviews([peAll]);

DECLARATION

```
property ReadOnly: Boolean;
```

DESCRIPTION

ReadOnly determines whether the user can change the contents of the field using the image control.

ImageEnDBView

DECLARATION

```
property StreamHeaders: Boolean
```

DESCRIPTION

If True (default), TImageEnDBView adds an additional header before standard image format.

To read older ImageEn data fields, leave this property as True (default value).

To read data field from other programs, set this property to False.

Events

DECLARATION

```
property OnBeforeLoadImage:
TIEBeforeLoadImageEvent;
```

DESCRIPTION

This event occurs just before an image is loaded.

Unit ImageENIO
TImageEnDBView

DECLARATION

```
property OnUnableToLoadImage:
TUnableToLoadImageEvent;
```

DESCRIPTION

This event occurs when TImageEnDBVect or TImageEnDBView fails to load image from blob field.

Chapter 24

TImageEnDBVect

Chapter 24. ImageEnDBVect

TImageEnDBVect is a descendant of TImageEnVect, but it can connected to a TDataset object to store/load images (BMP, PCX, GIF, JPEG, TIFF, PNG and others) and vectorial objects in Blob field.

It is similar to the TDBImage component of Delphi.
You can specify particular file format parameters through IOParams property, or executing DoIOPreview method.

You can attach a TImageEnProc component for image processing the image contained in the Blob.

Unit DBImageEnVect

TImageEnDBVect

Methods and Properties

AbsolutePath	AutoDisplay	DataField
DataFieldImageFormat	DataSource	DoIOPreview
Field	IOParams	IOPreviewsParams
JpegQuality	LoadedFieldImageFormat	LoadPicture
PreviewFont	ReadOnly	StreamHeaders

Events

OnUnableToLoadImage

Methods and Properties

DECLARATION

```
property AbsolutePath: string;
```

DESCRIPTION

The AbsolutePath property sets/gets the base path where the stored images are when DataField points to a string field.

The final path is calculated by concatenation of AbsolutePath and string field.

The default value is an empty string. If the AbsolutePath property is empty, then the string field should be an absolute path.

Unit DBImageEnVect
TImageEnDBVect

DECLARATION

```
property AutoDisplay: Boolean;
```

DESCRIPTION

AutoDisplay determines whether to automatically display the contents of a graphic BLOB in the database image control.

If AutoDisplay is True (the default value), the image automatically displays new data when the underlying BLOB field changes (such as when moving to a new record).

If AutoDisplay is False, the image clears whenever the underlying BLOB field changes.

To display the data, the user can double-click on the control or select it and press Enter. In addition, calling the LoadPicture method ensures that the control is showing data.

Change the value of AutoDisplay to False if the automatic loading of BLOB fields seems to take too long.

Unit DBImageEnVect
TImageEnDBVect

DECLARATION

```
property DataField: string;
```

DESCRIPTION

DataField specifies the field from which the database image displays data.

Use DataField to bind the image control to a field in the dataset. To fully specify a database field, both the dataset and the field within that dataset must be defined.

he DataSource property of the image control specifies the dataset which contains the DataField. DataField should specify a graphic field.

DECLARATION

```
property DataFieldImageFormat:
TDataFieldImageFormat;
```

DESCRIPTION

Sets the image format to save in the Blob field or path reference.

See the IOParams property for specific image format parameters.

Unit DBImageEnVect
TImageEnDBVect

DECLARATION

```
property DataSource: TDataSource;
```

DESCRIPTION

DataSource links the image control to a dataset.

Use DataSource to link the image control to a dataset in which the data can be found.

To fully specify a database field for the image control, both the dataset and a field within that dataset must be defined.

Use the DataField property to specify the particular field within the dataset.

DECLARATION

```
function DoIOPreview: Boolean;
```

DESCRIPTION

This function executes the IOPreviews dialog. This dialog gets/sets the parameters of image file formats.

The dialog shown depends on the DataFieldImageFormat property.

Unit DBImageEnVect

TImageEnDBVect

DECLARATION

```
property Field: TField;
```

DESCRIPTION

Field is the TField component the database image is linked to.

Read-only

DECLARATION

```
property IOParams: TIOParamsVals;
```

DESCRIPTION

IOParams allow you to set or get all file format parameters such as bits per pixel or type of compression.

Read-only

Example

```
Table1.Edit;
ImageEnDBView1.IOParams.BMP_Compression:=ioBMP_RL
E;
Table1.Post;
```

ImageEnDBVect

Unit DBImageEnVect

TImageEnDBVect

DECLARATION

```
property IOPreviewsParams: TIOPreviewsParams;
```

DESCRIPTION

This property contains some features that the input/output preview dialog will have.

DECLARATION

```
property JpegQuality: integer;
```

DESCRIPTION

JpegQuality sets the quality factor, from 1 to 100. The higher the value, the better the image quality and the larger resultant memory needed.

This is the example of IOParams.JPEG_Quality.

ImageEnDBVect

Unit DBImageEnVect
TImageEnDBVect

DECLARATION

```
function
LoadedFieldImageFormat():TDataFieldImageFormat;
```

DESCRIPTION

LoadedFieldImageFormat gets the image format stored in the Blob field (load directly from blob).

If you change DataFieldImageFormat, to store as several image formats, LoadedFieldImageFormat maintains the original value.

DECLARATION

```
procedure LoadPicture;
```

DESCRIPTION

LoadPicture loads the image stored in the field into the database image control.

Unit DBImageEnVect

TImageEnDBVect

DECLARATION

```
property PreviewFont: TFont;
```

DESCRIPTION

If PreviewFontEnabled is set to True then PreviewFont specifies the font used in the IOPreviews dialog. Ensure the size of font matches the labels length.

Example

```
ImageEnDBView1.PreviewFont.Name:='MS Times New
Roman';
ImageEnDBView1.PreviewFontEnabled := True;
ImageEnDBView1.DoPreviews([peAll]);
```

Unit DBImageEnVect

TImageEnDBVect

DECLARATION

```
property PreviewFontEnabled: TFont;
```

DESCRIPTION

By default (when PreviewFontEnabled = False) IO Preview dialogs are displayed using the system font (e.g. Tahoma or Segoe UI).
If you set PreviewFontEnabled to True then you can use PreviewFont to specify a custom font for the IO Preview dialogs.

Example

```
ImageEnDBView1.PreviewFont.Name:='MS Times New
Roman';
ImageEnDBView1.PreviewFontEnabled := False;
ImageEnDBView1.DoPreviews([peAll]);
```

DECLARATION

```
property ReadOnly: Boolean;
```

DESCRIPTION

ReadOnly determines whether the user can change the contents of the field using the image control.

Unit DBImageEnVect

TImageEnDBVect

DECLARATION

```
property StreamHeaders: Boolean
```

DESCRIPTION

If True (default), TImageEnDBView adds an additional header before standard image format.

To read older ImageEn data fields, leave this property as True (default value).

o read data field from other programs, set this property to False.

Events

DECLARATION

```
property OnUnableToLoadImage:
TUnableToLoadImageEvent;
```

DESCRIPTION

This event occurs when TImageEnDBVect or TImageEnDBView fails to load image from blob field.

Chapter 25

TIEBitmap

Chapter 25. IEBitmap

TIEBitmap is a replacement of VCL TBitmap class. It has many methods and properties compatible with TBitmap and enhances it supporting multi-threading and large images.

TIEBitmap can store images in memory mapped files (for big images), in memory (fast access) or can encapsulate TBitmap objects (for canvas drawing and compatibility).

Unit HYIEUtils

TIEBitmap

Methods and Properties

Access	Palette	PaletteLength
Allocate	Alpha	AlphaChannel
Assign	AssignImage	AutoCalcBWValues
BitAlignment	BitCount	BlackValue
CalcRAWSize	Canvas	CanvasCurrentAlpha
ChannelCount	ChannelOffset	Contrast
CopyAndConvertFormat	CopyFromMemory	CopyFromDIB
CopyFromTBitmap	CopyFromTDibBitmap	CopyFromTIEMask
CopyPaletteTo	CopyRectTo	CopyToTBitmap
CopyToTDibBitmap	CopyToTIEMask	CopyWithMask1
CopyWithMask2	Create	CreateDIB
DefaultDitherMethod	Destroy	DrawToCanvas
EncapsulatedFromMemory	EncapsulatedFromTBitmap	EncapsulateMemory
EncapsulateTBitmap	Fill	FillRect
FreeImage	FreeRow	Full
GetHash	GetRow	HasAlphaChannel
Height	IsAllBlack	IsEmpty
IsGrayScale	LoadRAWFromBuffer	Location
MemoryAllocator	MergeAlphaRectTo	MergeFromTDibBitmap

Unit HYIEUtils
TIEBitmap

MergeWithAlpha	MinFileSize	MoveRegion
Origin	PaletteUsed	PixelFormat
Pixels_ie16g	Pixels_ie1g	Pixels_ie24RGB
Pixels_ie32f	Pixels_ie32RGB	Pixels_ie48RGB
Pixels_ie8	Pixels_ieCIELab	Pixels_ieCMYK
Pixels	PPixels_ie24RGB	PPixels_ie32RGB
PPixels_ie48RGB	Read	RemoveAlphaChannel
RenderToCanvas	RenderToCanvasWithAlpha	RenderToTBitmap
RenderToTBitmapEx	RenderToTIEBitmap	RenderToTIEBitmapEx
Resize	Rowlen	SaveRAWToBufferOrStream
ScanLine	StretchRectTo	SwitchTo
SyncFull	SynchronizeRGBA	TBitmapScanlines
UndoInfo	UpdateFromTBitmap	VclBitmap
WhiteValue	Width	Write

Methods and Properties

DECLARATION

```
property Access: TIEDataAccess;
```

DESCRIPTION

Access specifies if the bitmap is readable or/and writable. This works only when Location is ieFile to speed up reading and writing operations.

The default value is [iedRead,iedWrite] that means read and write.

Unit HYIEUtils

TIEBitmap

DECLARATION

```
property Palette[index: integer]:TRGB;
```

DESCRIPTION

Palette returns the color of palette entry index

DECLARATION

```
property PaletteLength: integer;
```

DESCRIPTION

PaletteLength returns the palette entries count.

DECLARATION

```
function Allocate(ImageWidth, ImageHeight:
integer; ImagePixelFormat:
TIEPixelFormat=ie24RGB):Boolean;
```

DESCRIPTION

Allocate prepares space for an image with ImageWidth and ImageHeight sizes. When TIEBitmap is connected to TImageEnView, make sure that LegacyBitmap is False before set pixel formats other than ie1g and ie24RGB. Allocate returns true on successful.

IEBitmap

DECLARATION

```
property Alpha[x,y:integer]:byte;
```

DESCRIPTION

Alpha gets/sets one pixel alpha value. 0 is transparent, 255 is opaque. This function doesn't perform a range check test.

DECLARATION

```
property AlphaChannel: TIEBitmap;
```

DESCRIPTION

TIEBitmap handles the alpha channel as an encapsulated TIEBitmap object with pixelformat of ie8g. This property returns the associated AlphaChannel. If an image doesn't have an alpha channel, you can create it just using AlphaChannel property. To know if an image has an alpha channel, examine the HasAlphaChannel property.

DECLARATION

```
procedure Assign(Source: TObject);
```

DESCRIPTION

Assign copies an image from source object. Source can be a TIEBitmap or a TBitmap.

IEBitmap

DECLARATION

```
procedure AssignImage(Source: TIEBitmap);
```

DESCRIPTION

Assign copies an image from Source object, but not the alpha channel.

DECLARATION

```
procedure AutoCalcBWValues;
```

DESCRIPTION

Auto calculates BlackValue and WhiteValue to display correctly the image. This method finds the low and high values of image pixels and sets them to BlackValue and WhiteValue properties.

DECLARATION

```
property BitAlignment: integer;
```

DESCRIPTION

Specifies the number of bits of alignment (32 is the default). This works only when Location is ieMemory.

IEBitmap

Unit HYIEUtils
TIEBitmap

DECLARATION

```
property BitCount: integer;
```

DESCRIPTION

BitCount gets the bits per pixel. Here is the PixelFormat - BitCount comparison:
PixelFormat=ie1g -> BitCount=1
PixelFormat=ie8g -> BitCount=8
PixelFormat=ie8p -> BitCount=8
PixelFormat=ie16g -> BitCount=16
PixelFormat=ie24RGB -> BitCount=24
PixelFormat=ie32f -> BitCount=32
PixelFormat=ieCMYK -> BitCount=32
PixelFormat=ie48RGB -> BitCount=48
PixelFormat=ieCIELab -> BitCount=24
PixelFormat=ie32RGB -> BitCount=32

Changing the IEBitmap BitCount or Pixel Format

Convert IEBitmap to 32-bit

```
var
iBX, iBY; integer;
iRGB:  TRGB;
…
ImageEnView1.Proc.SaveUndo;
ImageEnView1.Proc.ClearAllRedo;
ImageEnView1.IO.Params.ImageIndex := 0;
// The Params.ImageIndex is valid for all file
// formats  (TIFF, GIF, DCX, etc..) and sets
// other properties like TIFF_ImageIndex,
// GIF_ImageIndex, etc. or you may use
// ImageEnView1.IO.Params.ICO_BitCount[ 0 ] := //
32;
// ImageEnView1.IO.Params.GIF_ImageIndex := 0;
// ImageEnView1.IO.Params.TIFF_ImageIndex := 0;
ImageEnView1.IO.Params.BitsPerSample := 8;
ImageEnView1.IO.Params.SamplesPerPixel := 4;
iBX := 0;
iBY := ImageEnView1.IEBitmap.Height - 1;
iRGB := ImageEnView1.IEBitmap.Pixels [ iBX, iBY
];
ImageEnView1.Proc.SetTransparentColors ( iRGB,
iRGB, 0 ); ImageEnView1.Update;
…
```

Note: You cannot undo IO.Params for display information purposes so to display the correct bitcount or pixel format you must save and restore the pixelformat separately using your own code for undo.

IEBitmap

Convert IEBitmap to 24-bit

```
…
ImageEnView1.Proc.SaveUndo;
ImageEnView1.Proc.ClearAllRedo;
ImageEnView1.IO.Params.BitsPerSample:= 8;
ImageEnView1.IO.Params.SamplesPerPixel := 3;
ImageEnView1.IO.Params.ImageIndex := 0;
// The Params.ImageIndex is valid for all file
// formats  (TIFF, GIF, DCX, etc.) and sets
// properties like TIFF_ImageIndex,
// GIF_ImageIndex, etc... or you may use
// ImageEnView1.IO.Params.ICO_BitCount [ 0 ] :=
// 24;
// ImageEnView1.IO.Params.GIF_ImageIndex := 0;
// ImageEnView1.IO.Params.TIFF_ImageIndex := 0;
if ImageEnView1.HasAlphaChannel then
  ImageEnView1.RemoveAlphaChannel;
ImageEnView1.Update;
…
```

Note: You cannot undo IO.Params for display information purposes so to display the correct bitcount or pixel format you must save and restore the pixelformat separately using your own code for undo.

Convert IEBitmap to 8-bit

```
…
ImageEnView1.Proc.SaveUndo;
ImageEnView1.Proc.ClearAllRedo;
ImageEnView1.IO.Params.BitsPerSample := 8;
ImageEnView1.IO.Params.SamplesPerPixel := 1;
ImageEnView1.IO.Params.ImageIndex := 0;
// The Params.ImageIndex is valid for all file
// formats  (TIFF, GIF, DCX, etc.) and sets
// properties like TIFF_ImageIndex,
// GIF_ImageIndex, etc... or you may use
// ImageEnView1.IO.Params.ICO_BitCount [ 0 ] :=
// 8;
// ImageEnView1.IO.Params.GIF_ImageIndex := 1;
// ImageEnView1.IO.Params.TIFF_ImageIndex := 0;
ImageEnView1.Proc.ConvertTo ( 256 );
ImageEnView1.Update;
…
```

Note: You cannot undo IO.Params for display information purposes so to display the correct bitcount or pixel format you must save and restore the pixelformat separately using your own code for undo.

IEBitmap

Convert IEBitmap to 4-bit

```
…
ImageEnView1.Proc.SaveUndo;
ImageEnView1.Proc.ClearAllRedo;
ImageEnView1.IO.Params.BitsPerSample := 4;
ImageEnView1.IO.Params.SamplesPerPixel := 1;
ImageEnView1.IO.Params.ImageIndex := 0;
// The Params.ImageIndex is valid for all file
// formats (TIFF, GIF, DCX, etc.) and sets
// properties like TIFF_ImageIndex,
// GIF_ImageIndex, etc… or you may use
// ImageEnView1.IO.Params.ICO_BitCount [ 0 ] :=
// 4;
// ImageEnView1.IO.Params.GIF_ImageIndex := 0;
// ImageEnView1.IO.Params.TIFF_ImageIndex := 0;
if ImageEnView.HasAlphaChannel then
  ImageEnView1.RemoveAlphaChannel;
ImageEnView1.Proc.ConvertTo ( 16 );
ImageEnView1.Update;
…
```

Note: You cannot undo IO.Params for display information purposes so to display the correct bitcount or pixel format you must save and restore the pixelformat separately using your own code for undo.

IEBitmap

Convert IEBitmap to 2-bit

```
…
ImageEnView1.Proc.SaveUndo;
ImageEnView1.Proc.ClearAllRedo;
ImageEnView1.IO.Params.BitsPerSample:= 2;
ImageEnView1.IO.Params.SamplesPerPixel:= 1;
ImageEnView1.IO.Params.ImageIndex := 0;
// The Params.ImageIndex is valid for all file
// formats (TIFF, GIF, DCX, etc.) and sets
// properties like TIFF_ImageIndex,
// GIF_ImageIndex, etc… or you may use
// ImageEnView1.IO.Params.ICO_BitCount [ 0 ] :=
// 2;
// ImageEnView1.IO.Params.GIF_ImageIndex := 0;
// ImageEnView1.IO.Params.TIFF_ImageIndex := 0;
if ImageEnView1.HasAlphaChannel then
  ImageEnView1.RemoveAlphaChannel;
ImageEnView1.Proc.ConvertTo ( 2 );
ImageEnView1.Update;
…
```

Note: You cannot undo IO.Params for display information purposes so to display the correct bitcount or pixel format you must save and restore the pixelformat separately using your own code for undo.

IEBitmap

DECLARATION

```
property BlackValue: double;
```

DESCRIPTION

BlackValue specifies (with WhiteValue) the values range starting from black to white.

For example, if your image is a gray scale (256 levels) where only values from 100 to 200 are used (100 is black and 200 is white), to display the image you have to write:

```
ImageEnView.IEBitmap.BlackValue:=100;
ImageEnView.IEBitmap.WhiteValue:=200;
ImageEnView.Update;
```

DECLARATION

```
function CalcRAWSize: integer;
```

DESCRIPTION

Calculates the space required for SaveRAWToBufferOrStream method, when used with 'Buffer" parameter.

IEBitmap

DECLARATION

```
property Canvas: TCanvas;
```

DESCRIPTION

Canvas property is available only when Location is ieTBitmap. Anyway accessing Canvas property Location is automatically converted to ieTBitmap.

DECLARATION

```
property CanvasCurrentAlpha: integer
```

DESCRIPTION

Setting this property to a value >=0, make the alpha channel an 8 bit gray scale paintable bitmap. This means that you can draw on the alpha channel using TCanvas object. The value (0..255) specifies the transparency, 0=fully transparent, 255=fully opaque.
The default is -1.

DECLARATION

```
property ChannelCount: integer;
```

DESCRIPTION

Returns number of channels of current pixel format.

DECLARATION

```
property ChannelOffset[idx: integer]: integer;
```

DESCRIPTION

ChannelOffset allows setting an offset for each channel. Idx is the channel where 0=red, 1=green and 2=blue. At the moment ChannelOffset works only with ie24RGB pixelformat.

For example, if you want to display only the red channel, just set green and blue to -255:

```
ImageEnView.IEBitmap.ChannelOffset[1]:=-255;
// hide green
ImageEnView.IEBitmap.ChannelOffset[2]:=-255;
// hide blue
ImageEnView.Update;
```

IEBitmap

ChannelOffset is useful also to increase or decrease luminosity (brightness). Example:

```
// trackbar1 has min=-255 and max=255.
ImageEnView.IEBitmap.ChannelOffset[0] :=
trackbar1.Position;
ImageEnView.IEBitmap.ChannelOffset[1] :=
trackbar1.Position;
ImageEnView.IEBitmap.ChannelOffset[2] :=
trackbar1.Position;
ImageEnView.Update;
```

Finally you can use ChannelOffset to display the alpha channel as a black image, hiding all channels. Example:

```
ImageEnView.IEBitmap.ChannelOffset[0] := -255;
ImageEnView.IEBitmap.ChannelOffset[1] := -255;
ImageEnView.IEBitmap.ChannelOffset[2] := -255;
ImageEnView.Update;
```

DECLARATION

```
property Contrast: integer;
```

DESCRIPTION

Contrast specifies a dynamic contrast to apply. It doesn't change the image but only how it is displayed.

Allowed values are 0 to 100.

IEBitmap

Unit HYIEUtils
TIEBitmap

DECLARATION

```
procedure CopyAndConvertFormat(Source:
TIEBitmap);
```

DESCRIPTION

Copies from specified source image converting to current pixel format. Uses color quantizers when convert from a true color to a paletted. ieTBitmap is not supported.

IEBitmap

DECLARATION

```
procedure CopyFromMemory(SrcBuffer: pointer;
SrcWidth: integer; SrcHeight: integer;
SrcPixelFormat: TIEPixelFormat; SrcOrigin:
TIEBitmapOrigin; SrcRowLen: integer);
```

DESCRIPTION

CopyFromMemory copies an image from memory buffer.

Parameter	Description
SrcBuffer	Source memory buffer.
SrcWidth	Source image width.
SrcHeight	Source image height.
SrcPixelFormat	Source pixel format.
SrcOrigin	Source orientation.
SrcRowLen	Source row length.

DECLARATION

```
procedure CopyFromDIB(Source: THandle);
procedure CopyFromDIB(BitmapInfo: pointer;
Pixels: pointer=nil);
```

DESCRIPTION

Copies from the specified DIB handle. The second overload copies from a DIB composed by a BitmapInfo structure and pixels buffer pointer. When Pixels is null, pixels are supposed to stay just after BitmapInfo structure.

DECLARATION

```
procedure CopyFromTBitmap(Source: TBitmap);
```

DESCRIPTION

CopyFromTBitmap gets the image from the Source TBitmap object. The source's TBitmap.PixelFormat allowed values are: pf1bit, pf4bit, pf8bit, pf15bit, pf16bit, pf24bit, pf32bit. If Location is ieTBitmap, accepts only pf1bit and pf24bit and pf32bit.

DECLARATION

```
procedure CopyFromTDibBitmap(Source:
TIEDibBitmap);
```

DESCRIPTION

CopyFromTDibBitmap gets the image from a TIEDibBitmap object.

DECLARATION

```
procedure CopyFromTIEMask(Source: TIEMask);
```

DESCRIPTION

CopyFromTIEMask gets the image from a TIEMask object.
If Source is nil then fills the bitmap with all 255.

DECLARATION

```
procedure CopyPaletteTo(Dest: TIEBaseBitmap);
```

DESCRIPTION

CopyPaletteTo copies all palette colors to destination bitmap. Dest must be a TIEBitmap object.

IEBitmap

DECLARATION

```
procedure CopyRectTo(Dest: TIEBitmap; SrcX, SrcY,
DstX, DstY: integer; RectWidth, RectHeight:
integer);
```

DESCRIPTION

CopyRectTo copies a rectangle to the Dest image. SrcX, SrcY, DstX, DstY specify the source and destination positions.
DstX and DstY can be negative (cut top-left rectangle and reduces size).
RectWidth and RectHeight specify the rectangle's width and height.
Dest must have same PixelFormat as the source image.
CopyRectTo doesn't copy the alpha channel.

DECLARATION

```
procedure CopyToTBitmap(Dest: TBitmap);
```

DESCRIPTION

CopyToTBitmap copies the image to the Dest TBitmap object.
These PixelFormat conversions are applied:
ie1g -> pf1bit
ie8p -> pf8bit
ie8g -> pf8bit (create gray scale palette)
ie16g -> pf8bit (copy only high 8 bit)
ie24RGB -> pf24bit

IEBitmap

Unit HYIEUtils

TIEBitmap

DECLARATION

```
procedure CopyToTDibBitmap(Dest: TIEDibBitmap;
source_x, source_y, sourceWidth,  sourceHeight:
integer);
```

DESCRIPTION

CopyToTDibBitmap copies specified source rectangle inside Dest (at top-left side). Dest.PixelFormat and PixelFormat must both be 1 bit or 24 bit.

DECLARATION

```
procedure CopyToTIEMask(Dest: TIEMask);
```

DESCRIPTION

CopyToIEMask copies the image to a TIEMask object.
It works only with ie1g and ie8g pixelformats. ie24RGB is converted to ie8g.

IEBitmap

DECLARATION

```
procedure CopyWithMask1(Dest: TIEBitmap;
SourceMask: TIEMask; Background: TColor);
procedure CopyWithMask1(Dest: TIEBitmap;
SourceMask: TIEMask);
```

DESCRIPTION

Copies current bitmap to Dest, using the SourceMask referred to the source.

DECLARATION

```
procedure CopyWithMask2(Dest: TIEBitmap;
DestMask: TIEMask);
```

DESCRIPTION

Copies current bitmap to Dest using DestMask referred to the destination.

IEBitmap

Unit HYIEUtils

TIEBitmap

DECLARATION

```
constructor Create;
constructor Create(ImageWidth, ImageHeight:
integer; ImagePixelFormat: TIEPixelFormat =
ie24RGB);
constructor Create(const FileName:string;
IOParams: TIOParamsVals = nil);
constructor Create(image: TIEBitmap);
```

DESCRIPTION

Create creates a new TIEBitmap object. The second overload creates the bitmap using specified parameters. Third overload loads image from specified file. IOParams specifies the in/out parameters as TIOParamsVals objects. Fourth overload creates a clone of source image.

DECLARATION

```
function CreateDIB: THandle;
function CreateDIB(x1,y1,x2,y2:integer):THandle;
```

DESCRIPTION

CreateDIB creates a DIB from current image. You are responsible to free memory (calling GlobalFree). The second overload creates a DIB from specified rectangle of current image.

IEBitmap

DECLARATION

```
property DefaultDitherMethod: TIEDitherMethod;
```

DESCRIPTION

DefaultDitherMethod specifies the default dithering method to apply when a color image needs to be converted to black/white. The default is ieThreshold.

DECLARATION

```
destructor Destroy;
```

DESCRIPTION

Destroy destroys this TIEBitmap object.

DECLARATION

```
procedure DrawToCanvas(DestCanvas: TCanvas;
xDst,yDst: integer);
```

DESCRIPTION

Draws the whole bitmap to the specified DestCanvas canvas, at xDst, yDst coordinates.

IEBitmap

DECLARATION

```
property EncapsulatedFromMemory: Boolean;
```

DESCRIPTION

EncapsulatedFromMemory is true if the image is encapsulated from a memory buffer (using EncapsulateMemory).

DECLARATION

```
property EncapsulatedFromTBitmap: Boolean;
```

DESCRIPTION

It is true if the image is encapsulated from a TBitmap object (using EncapsulateTBitmap).

Unit HYIEUtils
TIEBitmap

DECLARATION

```
procedure EncapsulateMemory(mem: pointer;
bmpWidth, bmpHeight: integer; bmpPixelFormat:
TIEPixelFormat; DoFreeImage: Boolean; Origin:
TIEBitmapOrigin = ieboBOTTOMLEFT);
```

DESCRIPTION

EncapsulateMemory encapsulates an existing bitmap from its buffer. It is useful to pass a TBitmap object to routines that require a TIEBitmap. If DoFreeImage is true, the buffer will be freed when the object is destroyed. Origin specifies the image orientation (default is bottom-left, that is Windows default. For example set ieboTOPLEFT with OpenCV images).

DECLARATION

```
procedure EncapsulateTBitmap(obj: TBitmap;
DoFreeImage: Boolean);
```

DESCRIPTION

EncapsulateTBitmap encapsulates an existing TBitmap object. It is useful to pass a TBitmap object to routines that require a TIEBitmap. If DoFreeImage is true, the TBitmap object will be freed when the object is destroyed. Supports only TBitmap with PixelFormat pf1bit or pf24bit

IEBitmap

DECLARATION

```
procedure Fill(Value:  double);
```

DESCRIPTION

Fill fills the image with a specified value. For ie1g images, Value can be 0 or 1. For ie8g and ie8p, it can be from 0 to 255. N For ie16g images, it can be from 0 to 65535. For ie24RGB, ie32RGB, ieCMYK and ieCIELab Value is a TColor value.

DECLARATION

```
procedure FillRect(x1,y1,x2,y2: integer; Value:
double);
```

DESCRIPTION

FillRect fills the rectangle with a specified value. For ie1g images, Value can be 0 or 1. For ie8g and ie8p, it can be from 0 to 255. For ie16g images, it can be from 0 to 65535. For ie24RGB, ie32RGB, ieCMYK and ieCIELab Value is a TColor value.

DECLARATION

```
procedure FreeImage(freeAlpha: Boolean = true);
```

DESCRIPTION

FreeImage destroys the image and frees memory.

DECLARATION

```
procedure FreeRow(Row: integer);
```

DESCRIPTION

FreeRow frees a row obtained using GetRow. FreeRow does nothing if Location is ieMemory or ieTBitmap.

DECLARATION

```
property Full: Boolean;
```

DESCRIPTION

Full is true all pixels are 1 (or white).
See also SyncFull.

DECLARATION

```
function GetHash(Algorithm: TIEHashAlgorithm =
iehaMD5): AnsiString;
```

DESCRIPTION

Calculates the hash (using the specified hash algorithm) of the bitmap and returns the string representation of the hash.

DECLARATION

```
function GetRow(Row: integer):pointer;
```

DESCRIPTION

GetRow gets a pointer to the specified Row. Returned pointer is valid until the application calls FreeRow. Just like Scanline if Location is ieMemory or ieTBitmap.

DECLARATION

```
property HasAlphaChannel:  Boolean;
```

DESCRIPTION

HasAlphaChannel specifies if the current image has an alpha channel.

DECLARATION

```
property Height: Integer;
```

DESCRIPTION

Height gets/sets the image height.

DECLARATION

```
function IsAllBlack:  Boolean;
```

DESCRIPTION

IsAllBlack returns true if all pixels are Zero.

DECLARATION

```
function IsEmpty:  Boolean;
```

DESCRIPTION

IsEmpty returns true if the bitmap is empty (width=0, height=0, etc...).

DECLARATION

```
function IsGrayScale:  Boolean;
```

DESCRIPTION

IsGrayScale checks if all pixels have R=G=B values (the bitmap is grayscale)

IEBitmap

Unit HYIEUtils

TIEBitmap

DECLARATION

```
function LoadRAWFromBufferOrStream(Buffer: pointer;  Stream: TStream): Boolean;
```

DESCRIPTION

LoadRAWFromBufferOrStream loads the image saved using SaveRAWToBufferOrStream.
Return true on success.

DECLARATION

```
property Location: TIELocation;
```

DESCRIPTION

Location specifies where store the image.

DECLARATION

TIELocation = (ieMemory, ieFile, ieTBitmap);

DESCRIPTION

Value	Description
ieMemory	Uses standard memory. Canvas not available. Used for fast and little images.
ieFile	Uses memory mapped files. Canvas not available. Used for big images.
ieTBitmap	Uses TBitmap VCL object. Canvas available. Used for drawing and compatibility. Location can be assigned before or after Allocate. Assigning Location on a existing image it converts the image to new location. ieTBitmap is a memory storage. Actually it uses the standard VCL TBitmap object as storage (hence it is the standard Windows DIB).

Unit HYIEUtils
TIEBitmap

DECLARATION

```
property MemoryAllocator: TIEMemoryAllocator;
```

DESCRIPTION

MemoryAllocator specifies which functions use to allocate bitmap memory.

DECLARATION

```
procedure MergeAlphaRectTo(Dest: TIEBitmap;
SrcX,  SrcY, DstX, DstY: integer; RectWidth,
RectHeight: integer);
```

DESCRIPTION

This method copies the alpha channel to Dest bitmap alpha channel.

DECLARATION

```
procedure MergeFromTDibBitmap(Source:
TIEDibBitmap; x,y: integer);
```

DESCRIPTION

MergeFromTDibBitmap gets the image from a TIEDibBitmap object placing it at the specified coordinates. It doesn't destroy original image. Source.PixelFormat and PixelFormat must be either 1 bit or 24 bit.

IEBitmap

DECLARATION

```
procedure MergeWithAlpha(Bitmap: TIEBitmap; DstX:
integer=0; DstY: integer=0; DstWidth: integer=-1;
DstHeight: integer=-1; Transparency: integer=255;
ResampleFilter: TResampleFilter = rfNone;
Operation: TIERenderOperation=ielNormal);
```

DESCRIPTION

MergeWithAlpha merges pixels and alpha channel of Bitmap with the background. MergeWithAlpha supports only ie24RGB pixel format.

Parameter	Description
Bitmap	Image with alpha channel to merge with the background.
DstX	Horizontal destination position.
DstY	Vertical destination position.
DstWidth	Destination width (Bitmap will be resampled to this value). -1 = the same width of source bitmap.
DstHeight	Destination height (Bitmap will be resampled to this value). -1 = the same height of source bitmap.
Transparency	Transparency of source bitmap. 0 = fully transparent, 255 = fully opaque.
ResampleFilter	Interpolation filter used when source bitmap needs to be resampled.
Operation	Blender operation to perform.

IEBitmap

DECLARATION

```
property MinFileSize:  int64;
```

DESCRIPTION

Specifies the minimum memory needed by the image to allow use of memory mapped file. If the memory needed by the image is less than MinFileSize, the image will be stored in memory (also if the Location is ieFile). If the global variable IEDefMinFileSize is not -1, it overlaps the property MinFileSize value.

DECLARATION

```
procedure MoveRegion(x1, y1, x2, y2, DstX, DstY:
integer; BackgroundValue: double; FillSource:
Boolean = true);
```

DESCRIPTION

MoveRegion moves a rectangle specified in x1, y1, x2, y2 to DstX, DstY position. The BackgroundValue parameter specifies the color that fills the source rectangle. If FillSource is true (default) then the source rectangle is filled with BackgroundValue. MoveRegion doesn't copy the alpha channel.

DECLARATION

```
property Origin: TIEBitmapOrigin;
```

DESCRIPTION

Origin specifies bitmap origin. Default is bottomleft that is Windows compatible. Other libraries could require topleft (like OpenCV).

DECLARATION

```
property PaletteUsed: integer;
```

DESCRIPTION

PaletteUsed specifies number of colors used by the palette.

DECLARATION

```
property PixelFormat: TIEPixelFormat;
```

DESCRIPTION

PixelFormat gets/sets the image pixelformat.
PixelFormat can convert image from a format to another.

DECLARATION

```
TIEPixelFormat = (ienull,
ie1g, // gray scale (black/white)
ie8p, // color (palette)
ie8g, // gray scale (256 levels)
ie16g, // gray scale (65536 levels)
ie24RGB, // RGB 24 bit (8 bit per channel)
ie32f, // float point values, 32 bit - Single in
Pascal - gray scale
ieCMYK, // CMYK (reversed 8 bit values)
ie48RGB, // RGB 48 bit (16 bit per channel)
ieCIELab, // CIELab (8 bit per channel)
ie32RGB // RGB 32 bit (8 bit per channel), last 8
bit are unused with some exceptions
);
```

DECLARATION

```
property Pixels_ie16g[x,y:integer]:word;
```

DESCRIPTION

Pixels_ie16g gets/sets one pixel value in 16 bit gray scale images (ie16g pixelformat). This function doesn't perform a range check test.

DECLARATION

```
property Pixels_ie1g[x,y: integer]: Boolean;
```

DESCRIPTION

Pixels_ie1g gets/sets one pixel value in black/white images (ie1g pixelformat). False is black, true is white. This function doesn't perform a range check test.

DECLARATION

```
property Pixels_ie24RGB[x,y:integer]:TRGB;
```

DESCRIPTION

Pixels_ie24RGB gets/sets one pixel value in true color images (ie24RGB pixelformat). This function doesn't perform a range check test.

DECLARATION

```
property Pixels_ie32f[x,y: integer]: single;
```

DESCRIPTION

Pixels_ie32f gets/sets one pixel value in 32 bit floating point value (single) for gray scale images (ie32f pixelformat). This function doesn't perform a range check test.

IEBitmap

DECLARATION

```
property Pixels_ie32RGB[x,y:integer]:TRGBA;
```

DESCRIPTION

Pixels_ie32RGB gets/sets one pixel value in true color images (ie32RGB pixelformat). This function doesn't perform a range check test.

DECLARATION

```
property Pixels_ie48RGB[x,y:integer]:TRGB48
```

DESCRIPTION

Pixels_ie48RGB gets/sets one pixel value in true color 48 bit images (ie48RGB pixelformat). This function doesn't perform a range check test.

DECLARATION

```
property Pixels_ie8[x,y:integer]:byte;
```

DESCRIPTION

Pixels_ie8 gets/sets one pixel value in palette or gray scale images (ie8p or ie8p pixelformat). This function doesn't perform a range check test.

IEBitmap

DECLARATION

```
property Pixels_ieCIELab[x,y:integer]: TCIELab
```

DESCRIPTION

Pixels_ieCIELab gets/sets one pixel value in CIELab color space (ieCIELab pixelformat). This function doesn't perform a range check test.

DECLARATION

```
property Pixels_ieCMYK[x,y:integer]: TCMYK;
```

DESCRIPTION

Pixels_ieCMYK gets or sets one pixel value in CMYK true color images (ieCMYK pixelformat). This function doesn't perform a range check test.

DECLARATION

```
property Pixels[x,y:integer]: TRGB;
```

DESCRIPTION

Pixels get the RGB value for the specified pixel. For black/white images it can be (0, 0, 0) or (255, 255, 255). This function doesn't perform a range check test.

DECLARATION

```
property PPixels_ie24RGB[x, y: integer]:  PRGB;
```

DESCRIPTION

PPixels_ie24RGB returns a pointer to a specified pixel.
This function doesn't perform a range check test.

DECLARATION

```
property PPixels_ie32RGB[x, y: integer]: PRGBA;
```

DESCRIPTION

PPixels_ie32RGB Returns a pointer to a specified pixel.
This function doesn't perform a range check test.

DECLARATION

```
property PPixels_ie48RGB[x, y: integer]: PRGB48;
```

DESCRIPTION

PPixels_ie48RGB returns a pointer to a specified pixel.
This function doesn't perform a range check test.

DECLARATION

```
function Read(const FileName:string; IOParams:
TObject = nil):Boolean; overload;
function Read(Stream: TStream; IOParams: TObject
= nil):Boolean; overload;
```

DESCRIPTION

Read loads image from file or stream. This method supports all formats supported by TImageEnIO class. IOParams specifies the in/out parameters as TIOParamsVals objects. Read returns false on failure.

DECLARATION

```
procedure RemoveAlphaChannel(Merge:
Boolean=false; BackgroundColor:TColor = clWhite);
```

DESCRIPTION

RemoveAlphaChannel removes the associated alpha channel. When Merge is true, the specified BackgroundColor is merged with the semitransparent areas of the image (like a shadow).

DECLARATION

```
procedure RenderToCanvas(DestCanvas: TCanvas;
xDst,yDst,dxDst,dyDst: integer; Filter:
TResampleFilter; Gamma: double);
```

DESCRIPTION

Draws the rectangle specified by xDst, yDst, dxDst, dyDst to DestCanvas. Filter specifies the filter if the image needs to be resampled. Gamma is the gamma correction.

DECLARATION

```
procedure RenderToCanvasWithAlpha(Dest: TCanvas;
xDst,yDst,dxDst,dyDst: integer; xSrc, ySrc,
dxSrc, dySrc: integer; Transparency: integer;
Filter: TResampleFilter; RenderOperation:
TIERenderOperation);
```

DESCRIPTION

RenderToCanvasWithAlpha draws the rectangle xSrc, ySrc, dxSrc, dySrc inside the destination rectangle xDst, yDst, dxDst, dyDst of Dest TCanvas object. Transparency specifies the transparency value (0 to 255). Filter the resampling filter. RenderOperation is the rendering operation.

This function reads the destination canvas pixels and merges them with image using alpha channel mask.

IEBitmap

Unit HYIEUtils
TIEBitmap

DECLARATION

```
procedure RenderToTBitmap(ABitmap: TBitmap; var
ABitmapScanline: ppointerarray; var XLUT, YLUT:
pinteger;  UpdRect:  PRect; xDst, yDst, dxDst,
dyDst: integer; xSrc, ySrc, dxSrc, dySrc:
integer; EnableAlpha: Boolean; SolidBackground:
Boolean; Transparency: integer; Filter:
TResampleFilter; FreeTables: Boolean;
RenderOperation: TIERenderOperation);
```

DESCRIPTION

RenderToTBitmap is used internally to render a TIEBitmap in TImageEnView object. ABitmap must be pf24bit

Unit HYIEUtils
TIEBitmap

DECLARATION

```
procedure RenderToTIEBitmap(ABitmap: TIEBitmap;
var ABitmapScanline: ppointerarray; var XLUT,
YLUT: pinteger; UpdRect: PRect; xDst, yDst,
dxDst, dyDst: integer; xSrc, ySrc, dxSrc, dySrc:
integer; EnableAlpha: boolean; SolidBackground:
boolean; Transparency: integer; Filter:
TResampleFilter; FreeTables: boolean;
RenderOperation: TIERenderOperation);
```

DESCRIPTION

RenderToTIEBitmap is used internally to render a TIEBitmap in a TImageEnView object.

ABitmap must be 24 bit.
ABitmapScanline can be nil
XLUT, YLUT: x and y screen to bitmap conversion table, can be NIL.

IEBitmap

Unit HYIEUtils
TIEBitmap

Declaration

```
procedure RenderToTBitmapEx(Dest: TBitmap; xDst,
yDst, dxDst, dyDst: integer; xSrc, ySrc, dxSrc,
dySrc: integer; Transparency: integer; Filter:
TResampleFilter; RenderOperation:
TIERenderOperation);
```

Description

RenderToTBitmapEx draws the rectangle xSrc, ySrc, dxSrc, dySrc inside the destination rectangle xDst, yDst, dxDst, dyDst of Dest TBitmap object.

Dest.PixelFormat must be pf24bit.
Transparency specifies the transparency value (0 to 255).
Filter the resampling filter.
RenderOperation is the rendering operation.

IEBitmap

Unit HYIEUtils
TIEBitmap

Declaration

```
procedure RenderToTIEBitmapEx(Dest: TIEBitmap;
xDst, yDst, dxDst, dyDst: integer; xSrc, ySrc,
dxSrc, dySrc: integer; Transparency: integer;
Filter: TResampleFilter; RenderOperation:
TIERenderOperation);
```

Description

RenderToTIEBitmapEx draws the rectangle xSrc, ySrc, dxSrc, dySrc inside the destination rectangle xDst, yDst, dxDst, dyDst of Dest TIEBitmap object.

Dest.PixelFormat must be ie24RGB.
Transparency specifies the transparency value (0 to 255).
Filter the resampling filter.
RenderOperation is the rendering operation.

IEBitmap

DECLARATION

```
procedure Resize(NewWidth, NewHeight: integer;
BackgroundValue: double; AlphaValue: integer;
HorizAlign: TIEHAlign; VertAlign: TIEVAlign);
```

DESCRIPTION

Resize resizes the image to specified NewWidth and NewHeight values without resampling the image. This method resizes also the alpha channel.

Parameter	Description
NewWidth	New image width.
NewHeight	New image height.
BackgroundValue	TColor value for ie24RGB images or a gray level for gray scale or black/white images. It is used to fill added regions.
AlphaValue	Alpha value used to fill added regions.
HorizAlign	Specifies how to horizontally align the old image.
VertAlign	Specifies how to vertically align the old image.

Unit HYIEUtils

TIEBitmap

DECLARATION

```
property RowLen: integer;
```

DESCRIPTION

RowLen gets the count of bytes in a row. For default a row is double word aligned (32bit).

DECLARATION

```
procedure SaveRAWToBufferOrStream(Buffer:
pointer; Stream: TStream);
```

DESCRIPTION

SaveRAWToBufferOrStream saves the image as internal format preserving the pixel format and alpha channel. Location field will be lost. Only the image and its description is saved. If the image has alpha channel this is also saved. You can save the image inside a buffer or a stream. Saving in a buffer you have to specify a valid Buffer (of size returned by CalcRAWSize) and Stream must be nil. Saving in a Stream the Buffer parameter must be nil.

Look also

LoadRAWFromBufferOrStream

CalcRAWSize

IEBitmap

DECLARATION

```
property Scanline[row: integer]:pointer
```

DESCRIPTION

Scanline gets an entire line of pixels at a time.
For Location=ieFile you can use only last line obtained from Scanline[].

DECLARATION

```
procedure StretchRectTo(Dest: TIEBitmap; xDst,
yDst, dxDst, dyDst: integer; xSrc, ySrc, dxSrc,
dySrc: integer; Filter: TResampleFilter;
Transparency: integer);
```

DESCRIPTION

Stretches source rectangle in destination rectangle. This method doesn't merge the image with the background, but just replace it (image and alpha). B This function assumes that there is an alpha channel (if not creates it). Dest must be ie24RGB

DECLARATION

```
procedure StretchValues();
```

DESCRIPTION

StretchValues stretches pixels value in the range of BlackValue and WhiteValue. This method resets BlackValue and WhiteValue to zero. StretchValues works only when PixelFormat is ie8g, ie16g, ie24RGB and ie32f.

DECLARATION

```
procedure SwitchTo(Target: TIEBitmap);
```

DESCRIPTION

SwitchTo assigns current image to the target object. The source object will be empty. SwitchTo executes more quickly than a copy operation because the image is transferred not copied. In detail it assigns all fields to Target, and set to zero all parameters (do not free the image or allocated memory).

DECLARATION

```
procedure SyncFull;
```

DESCRIPTION

SyncFull sets Full to true if all values of the image are 255.

DECLARATION

```
procedure SynchronizeRGBA(RGBAtoAlpha: Boolean);
```

DESCRIPTION

When ie32RGB (RGBA) pixel format is used, the A channel is not used. Alpha channel is stored in a separated plane (AlphaChannel). To copy from A channel to Alpha channel you have to call SynchronizeRGBA with RGBAToAlpha=true. Vice versa to copy from Alpha channel to A channel call SynchronizeRGBA with RGBAToAlpha=false.

DECLARATION

```
property TBitmapScanlines:  ppointerarray;
```

DESCRIPTION

TBitmapScanlines returns an array of pointers for each scanline of the bitmap.

DECLARATION

```
property UndoInfo: pointer;
```

DESCRIPTION

Used internally to store undo information.

DECLARATION

```
procedure UpdateFromTBitmap;
```

DESCRIPTION

UpdateFromTBitmap updates TIEBitmap fields to the content of embedded TBitmap object (VclBitmap). This is useful if you change VclBitmap and wants to refresh TIEBitmap.

DECLARATION

```
property VclBitmap: TBitmap;
```

DESCRIPTION

VclBitmap contains the encapsulated TBitmap object.

IEBitmap

Unit HYIEUtils
TIEBitmap

DECLARATION

```
property WhiteValue: double;
```

DESCRIPTION

WhiteValue specifies (with BlackValue) the values range starting from black to white.

For example, if your image is a gray scale (256 levels) where only values from 100 to 200 are used (100 is black and 200 is white), to display the image you have to write:

```
ImageEnView.IEBitmap.WhiteValue := 100;
ImageEnView.Update;
```

DECLARATION

```
property Width: Integer;
```

DESCRIPTION

Width gets/sets the image width.

DECLARATION

```
procedure Write(const FileName:string; IOParams:
TObject = nil); overload;
procedure Write(Stream: TStream; FileType:
integer;  IOParams: TObject = nil); overload; //
FileType is TIOFileType
```

DESCRIPTION

Writes image to file or stream. This method supports all formats supported by TImageEnIO class. IOParams specifies the in/out parameters as TIOParamsVals objects.

IEBitmap

Unit HYIEUtils
TIEBitmap

DECLARATION

```
procedure Write(const FileName: string; IOParams:
TObject = nil); overload;
procedure Write(Stream: TStream; FileType:
integer; IOParams: TObject = nil); overload; //
FileType is TIOFileType
```

DESCRIPTION

Writes image to file or stream. This method supports all formats supported by TImageEnIO class. IOParams specifies the in/out parameters as TIOParamsVals objects.

Examples

```
var
bmp: TIEBitmap;
begin
  bmp := TIEBitmap.Create;
  bmp.Read('input.jpg');
  bmp.Write('output.jpg');
  bmp.Free;
end;
```

...that is the same as...

```
with TIEBitmap.Create('input.jpg') do
begin
 Write('output.jpg');
 Free;
end;
```

IEBitmap

...that is the same as...

```
var
bmp: TIEBitmap;
io: TImageEnIO;
begin
  bmp := TIEBitmap.Create;
  io := TImageEnIO.CreateFromBitmap(bmp);
  io.LoadFromFile('input.jpg');
  io.SaveToFile('output.jpg');
  io.Free;
  bmp.Free;
end;
```

Chapter 26

TIEMask

Chapter 26. IEMask

TIEMask contains a selection, which is a map of selected and unselected pixels. A selection map can have a depth of 1 bit or 8 bit.

For a map of 1 bit, 0 is a non-selected pixel, while 1 is selected. For a map of 8 bit, 0 is non-selected pixel, >0 is “semi” selected pixel up to 255 that means fully selected pixel.

TImageEnView component uses this class to store current selection in SelectionMask property.

Unit HYIEUtils

TIEMask

Methods and Properties

AllocateBits	Assign	Bits
BitsPerPixel	CombineWithAlpha	CopyBitmap
CopyIEBitmap	CopyIEBitmapAlpha	DrawOuter
DrawOutline	DrawPolygon	Empty
Fill	FreeBits	Full
GetPixel	Height	Intersect
InvertCanvas	IsEmpty	IsPointInside
Negative	Resize	Rowlen
ScanLine	SetPixel	SyncFull
SyncRect	TranslateBitmap	Width
X1	X2	Y1
Y2		

Declaration

```
procedure AllocateBits(width, height: integer;
bitsperpixel: integer);
```

Description

AllocateBits allocates memory for the map. Memory is zero filled. Bitsperpixel can be 1 or 8.

Declaration

```
procedure Assign(Source: TIEMask);
```

Description

Assign assigns Source mask.

Declaration

```
property Bits: pbyte;
```

Description

Bits contain the raw buffer of the selection mask.

Declaration

```
property BitsPerPixel:  integer;
```

Description

Bits per pixels of the mask. ImageEn supports only 1 bit selection masks (1 selected, 0 not selected).

ReadOnly

Declaration

```
procedure CombineWithAlpha(SourceAlpha:
TIEBitmap; ox, oy: integer;
SynchronizeBoundingRect: Boolean);
```

Description

Only for internal use.

Declaration

```
procedure CopyBitmap(Dest, Source: TBitmap;
dstonlymask, srconlymask:  Boolean);
```

Description

For internal use only.

Declaration

```
procedure CopyIEBitmap(Dest, Source: TIEBitmap;
dstonlymask, srconlymask: Boolean;
UseAlphaChannel: Boolean);
```

Description

For internal use only.

Declaration

```
procedure CopyIEBitmapAlpha(Dest, Source:
TIEBitmap; dstonlymask, srconlymask: Boolean);
```

Description

For internal use only.

Declaration

```
procedure DeleteImage(imageIndex: integer);
```

Description

DeleteImage removes the specified image.

Declaration

```
procedure DrawOuter(Dest: TBitmap; xDst, yDst,
dxDst, dyDst: integer; xMask, yMask, dxMask,
dyMask: integer; AlphaBlend: integer; Color:
TColor);
```

Description

For internal use only.

Declaration

```
procedure DrawOutline(Dest: TCanvas; xDst, yDst,
dxDst, dyDst: integer; xMask, yMask, dxMask,
dyMask: integer; AniCounter: integer; Color1,
Color2: TColor; actualRect: PRect = nil);
```

Description

For internal use only.

Declaration

```
procedure DrawPolygon(Alpha: integer; SelPoly:
PPointArray; SelPolyCount: integer);
```

Description

Draw specified polygon in the mask, using Alpha value for all pixels.

IEMask

Declaration

```
procedure Empty;
```

Description

Empty fills the entire mask with zeros.

Declaration

```
procedure Fill(Alpha: integer);
```

Description

Fills the entire mask using Alpha value.

Declaration

```
procedure FreeBits;
```

Description

Free the allocated mask.

Declaration

```
property Full: Boolean;
```

Description

Fill is true when the mask contains all 1 values (that is the image has all pixels selected).

DECLARATION

```
function GetPixel(x, y:integer): integer;
```

DESCRIPTION

GetPixel returns the value of the pixel at x,y coordinates.

DECLARATION

```
property Height: integer;
```

DESCRIPTION

Height of the selection mask. It must be equal to the image height.

ReadOnly

DECLARATION

```
procedure InsertImage(imageIndex: integer);
```

DESCRIPTION

InsertImage inserts a new image at the specified position.

DECLARATION

```
procedure Intersect(x1, y1, x2, y2: integer);
```

DESCRIPTION

For internal use only.

DECLARATION

```
procedure InvertCanvas(Dest: TCanvas; xDst, yDst,
dxDst, dyDst: integer; xMask, yMask, dxMask,
dyMask: integer);
```

DESCRIPTION

For internal use only.

DECLARATION

```
function IsEmpty: Boolean;
```

DESCRIPTION

Return True is the mask contains all zeros.

IEMask

Unit HYIEUtils

TIEMask

Declaration

```
function IsPointInside(x,y:integer):Boolean;
```

Description

Return true if the x,y pixel is not 0.

Declaration

```
procedure Negative(MaxVal: integer);
```

Description

For internal use only.

Declaration

```
procedure Resize(NewWidth, NewHeight: integer);
```

Description

Resize resizes this selection map.

Declaration

```
property Rowlen: integer;
```

Description

Rowlen returns the length of a row in bytes.
ReadOnly

IEMask

Declaration

```
property ScanLine[row: integer]: pointer;
```

Description

Like TBitmap.Scanline property. It allows getting or setting mask by hand.

Declaration

```
procedure SetPixel(x,y: integer; Alpha: integer);
```

Description

SetPixel sets a single pixel selection at x,y coordinates. A pixel with mask 1 or >0 is selected.

Declaration

```
procedure SyncFull;
```

Description

SyncFull sets Full to true if all values are 255.

DECLARATION

```
procedure SyncRect;
```

DESCRIPTION

SyncRect adjusts X1, Y1, X2 and Y2 to enclose the bounding box of the selected pixels.

DECLARATION

```
procedure TranslateBitmap(var ox, oy: integer;
CutSel: Boolean);
```

DESCRIPTION

TranslateBitmap translates the mask of ox, oy pixels. CutSel is true if the mask can go out of mask margins.

DECLARATION

```
property Width: integer;
```

DESCRIPTION

Width is the width of the selection mask. It must be equal to the image width.

ReadOnly

IEMask

Declaration

```
property X1: integer;
```

Description

X1 is the left-up side of the non-empty selection (an empty mask has 1).

Declaration

```
property X2: integer;
```

Description

X2 is the right-bottom side of the non-empty selection.

Declaration

```
property Y1: integer;
```

Description

Y1 is the left-up side of the non-empty selection (an empty mask has 1).

IEMask

DECLARATION

```
property Y2: integer;
```

DESCRIPTION

Y2 is the right-bottom side of the non-empty selection.

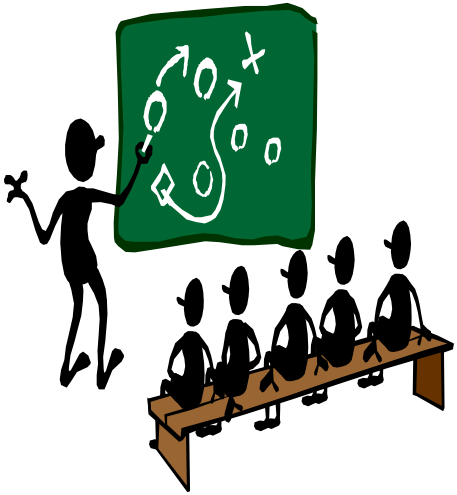
Chapter 27

TIEAnimation

Chapter 27. IEAnimation

Declaration

```
TIEAnimation = class;
```



Description

TIEAnimation is the base abstract class for animation classes like coverflow-like (TIEHorizontalFlow) and circular cflow (TIECircularFlow) or any user custom animation classes).

Unit HYIEUtils

TIEAnimation

Methods and Properties

Protected

DoGetImage	DoGetImageInfo	DoReleaseImage
DrawImage **Virtual**	PaintScrollBar	PaintText
SetEndValues **Virtual**	SetInitialValues **Virtual**	SetStartEndValues
SetStartValues **Virtual**		

Public

Constructor Create	**Destructor** Destroy	AnimDuration
CurrentImage	Depth	Display **Virtual**
FindImageAt	FramesZoomFilter	ImageCount
Images	IsInsideScrollbar	IsInsideScrollbarSlider
LastFrameZoomFilter	MoveScrollbarSliderTo	NeedRefresh
RestartAnimation	SetupEvents	ShadowAlphaMax
ShadowAlphaMin	ShowBorder	ShowScrollbar
ShowText	ViewHeight	ViewWidth

Unit HYIEUtils
TIEAnimation

DECLARATION

```
procedure DoGetImage(imageIndex: integer; var
image: TIEBitmap; var text: WideString);
```

DESCRIPTION

DoGetImage calls the TIEAnimationGetImageEvent event, assigned on creation.

Parameter	Description
imageIndex	Index of the required image.
image	Return value for image bitmap.
text	Return value for image textual info (ie a filename).

TIEAnimation Links

Unit HYIEUtils
TIEAnimation

DECLARATION

```
procedure DoGetImageInfo(imageIndex: integer;
isVisible: Boolean; var imageWidth: integer; var
imageHeight: integer; var text: WideString);
```

DESCRIPTION

DoGetImageInfo calls the TIEAnimationGetImageInfoEvent event, if assigned on creation. If no event was assigned just calls the TIEAnimationGetImageEvent event.

Parameter	Description
imageIndex	Index of the required image info.
isVisible	Is True if the required image could be not visible (hence you can provide estimated values).
imageWidth	Return value for image width.
imageHeight	Return value for image height
text	Return value for image textual info (ie a filename).

Unit HYIEUtils
TIEAnimation

DECLARATION

```
procedure DoReleaseImage(imageIndex: integer; var image: TIEBitmap);
```

DESCRIPTION

DoReleaseImage calls the TIEAnimationReleaseImageEvent event, if assigned on creation.

Parameter	Description
imageIndex	Index of the image to release.
image	The bitmap to release.

DECLARATION

```
function DrawImage(dest: TIEBitmap; imageIndex:
integer):Boolean; virtual;
```

DESCRIPTION

DrawImage draws the specified image. This method reads the timers to draw the image correctly (at calculated position, rotation, etc...). Returns true when the image doesn't need to be drawn again (the animation is finished).

Parameter	Description
dest	The destination bitmap. Must be at least ViewWidth and ViewHeight pixels.
imageIndex	The image index to draw.

DECLARATION

```
procedure PaintScrollBar(dest: TIEBitmap);
```

DESCRIPTION

PaintScrollBar paints the scroll bar. You should call this method inside Display method.

Parameter	Description
dest	The destination bitmap.

Declaration

```
procedure PaintText(posY: integer; dest:
TIEBitmap);
```

Description

PaintText paints the text (ie filename) at the specified row. You should call this method inside Display method.

Parameter	Description
posY	Vertical position where to draw the text.
dest	The destination bitmap.

Declaration

```
procedure SetEndValues(); virtual;
```

Description

SetEndValues sets ending values (image position, rotation, etc.) whenever the animation is about to begin. This method needs to be overloaded to implement animations.

DECLARATION

```
procedure SetInitialValues(); virtual;
```

DESCRIPTION

Sets initial animation values (image position, rotation, etc.). This is called one time when the object is initialized. This method need to be overloaded to implement animations.

DECLARATION

```
procedure SetStartEndValues();
```

DESCRIPTION

This method just calls SetStartValues and SetEndValues.

DECLARATION

```
procedure SetStartValues(); virtual;
```

DESCRIPTION

SetStartValues sets starting values (image position, rotation, etc.) whenever the animation is about to begin. This method needs to be overloaded to implement animations.

Unit HYIEUtils
TIEAnimation

DECLARATION

```
constructor Create();
```

DESCRIPTION

Create creates the TIEAnimation object. You should call this constructor on your subclass.

DECLARATION

```
destructor Destroy(); override;
```

DESCRIPTION

Destroy destroys the TIEAnimation object. You should call this destructor on your subclass.

Declaration

```
property AnimDuration: dword;
```

Description

AnimDuration specifies the animation duration in milliseconds. Default is 800ms.

Declaration

```
property CurrentImage: integer;
```

Description

CurrentImage specifies current image index. Current image is often the most visible or the central image.

Declaration

```
property Depth: integer;
```

Description

Depth specifies the 3D projection depth. Default is 500.

DECLARATION

```
procedure Display(dest: TIEBitmap); virtual;
abstract;
```

DESCRIPTION

This method must be implemented by the subclass. It should loop among images calling drawImage for each image.

Parameter	Description
dest	The destination bitmap.

DECLARATION

```
function FindImageAt(X, Y: integer): integer;
```

DESCRIPTION

Returns the index of the image that is under the specified coordinates or 1 if there is no image.

Parameter	Description
X	Horizontal coordinate.
Y	Vertical coordinate.

DECLARATION

```
property FramesZoomFilter: TResampleFilter;
```

DESCRIPTION

FramesZoomFilter specifies the interpolation filter to apply when the animation runs. Default is rfNone.

DECLARATION

```
property ImageCount: Integer;
```

DESCRIPTION

ImageCount specifies the number of images that TIEAnimation should handle. This value could not match the number of visible images. Setting this property reinitializes the animation and set the current image to 0.

DECLARATION

```
property Images[index: integer]:
TIEAnimationImageInfo;
```

DESCRIPTION

Images allows getting or setting animation parameters for the specified image index.

Unit HYIEUtils
TIEAnimation

DECLARATION

```
TIEAnimationImageInfo = class;
```

DESCRIPTION

Objects of this class contain the animation state of each image like positions and rotations, other than timing info.

Fields

```
TIEAnimationImageInfo = class
public

startTime :  dword; // when transition was started
endTime :  dword; // when transition must ends
startAlpha :  integer; // the starting alpha (0..255)
endAlpha :  integer; // the ending alpha (0..255)
lastAlpha :  integer; // last calculated alpha (0..255)
startAngleX :  double; // the starting angle X in degrees
endAngleX :  double; // the ending angle X in degrees
lastAngleX :  double; // last calculated angle X in degrees
startAngleY :  double; // the starting angle Y in degrees
endAngleY :  double; // the ending angle Y in degrees
lastAngleY :  double; // last calculated angle Y in degrees
startCenterX :  integer; // the starting X position
endCenterX :  integer; // the ending X position
lastCenterX :  integer; // last calculated X position
startCenterY :  integer; // the starting Y position
endCenterY :  integer; // the ending Y position
lastCenterY :  integer; // last calculated Y position
startWidth :  integer; // the starting width
endWidth :  integer; // the ending width
lastWidth :  integer; // last calculated width
startHeight : integer; // the starting height
endHeight :  integer; // the ending height
lastHeight :  integer; // last calculated height
lastCoords : TIEQuadCoords; // last calculated coordinates
end;
```

Unit HYIEUtils
TIEAnimation

DECLARATION

```
function IsInsideScrollbar(X, Y: integer):
Boolean;
```

DESCRIPTION

IsInsideScrollbar returns true if the specified coordinates are inside the scrollbar.

Parameter	Description
X	Horizontal coordinate.
Y	Vertical coordinate.

DECLARATION

```
function IsInsideScrollbarSlider(X, Y:integer): Boolean;
```

DESCRIPTION

IsInsideScrollbarSlider returns true if the specified coordinates are inside the scrollbar slider. This function is useful to start dragging the scrollbar slider.

Parameter	Description
X	Horizontal coordinate.
Y	Vertical coordinate.

DECLARATION

```
property LastFrameZoomFilter: TResampleFilter;
```

DESCRIPTION

LastFrameZoomFilter specifies the interpolation filter to apply when the last frame is showed (when the animation ends). Default is rfFastLinear.

DECLARATION

```
procedure MoveScrollbarSliderTo(X: integer);
```

DESCRIPTION

MoveScrollbarSliderTo moves the scrollbar slider to the specified horizontal position.

Parameter	Description
X	Horizontal slider position.

DECLARATION

```
property NeedRefresh: Boolean;
```

DESCRIPTION

Applications will read this value to know if a call to Display is necessary. This property is true when animation is running and false when the last frame has been painted. NeedRefresh can be also true just after a property has been changed.

TIEAnimation

Declaration

```
procedure RestartAnimation();
```

Description

RestartAnimation restarts the animation.

Declaration

```
procedure SetupEvents(getImage:
TIEAnimationGetImageEvent; releaseImage:
TIEAnimationReleaseImageEvent = nil;
getImageInfo: TIEAnimationGetImageInfoEvent =
nil);
```

Description

Setup events are necessary to show images.

Parameter	Description
getImage	The event called whenever a new image needs to be drawn. Cannot be nil.
releaseImage	The event called whenever an image can be released. Can be nil.
getImageInfo	The event called whenever only image info are needed. Can be nil.

TIEAnimation

Declaration

```
property ShadowAlphaMax: integer;
```

Description

ShadowAlphaMax specifies the maximum alpha value for the shadow. Default is 40.

Declaration

```
property ShadowAlphaMin: integer;
```

Description

ShadowAlphaMin specifies the minimum alpha value for the shadow. Default is 0.

Declaration

```
property ShowBorder:  Boolean;
```

Description

If true a border will be painted around all images.

DECLARATION

```
property ShowScrollbar: Boolean;
```

DESCRIPTION

If true the scrollbar will be painted.

DECLARATION

```
property ShowText: Boolean;
```

DESCRIPTION

If true the current image text will be painted.

DECLARATION

```
property ViewHeight: integer;
```

DESCRIPTION

Specifies the height of area reserved for the animation.

See also

ViewWidth

Unit HYIEUtils

TIEAnimation

DECLARATION

```
property ViewWidth: integer;
```

DESCRIPTION

ViewWidth specifies the width of area reserved for the animation.

See also

ViewHeight

Demos

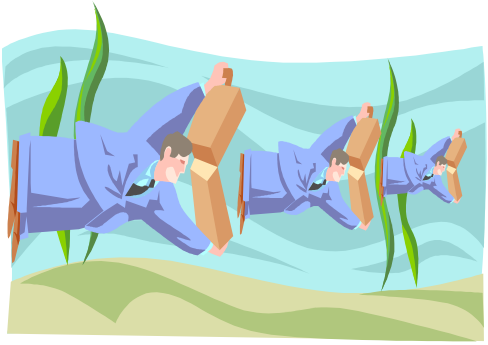
viewers\manualFlow

Chapter 28

TIEHorizontalFlow

Chapter 28. IEHorizontalFlow

TIEHorizontalFlow = class(TIEAnimation);



DESCRIPTION

The TIEHorizontalFlow class implements a coverflow-like animation.

Methods and Properties

CurrentImageZoom

HorizontalDistance

ImagesHorizontalPercentage

ImagesVerticalPercentage

ImagesZoom

RotateAngle

Methods and Properties

DECLARATION

```
property CurrentImageZoom: double;
```

DESCRIPTION

Specifies the zoom (0..1) for central image. Default is 1.0.

DECLARATION

```
property HorizontalDistance: integer;
```

DESCRIPTION

HorizontalDistance specifies the horizontal distance among images, in pixels. Default is 80.

DECLARATION

```
property ImagesHorizontalPercentage: double;
```

DESCRIPTION

Specifies the percentage (1..100) of view-width used by a single image. Default is 40.

DECLARATION

```
property ImagesVerticalPercentage: double;
```

DESCRIPTION

ImagesVerticalPercentage specifies the percentage (1..100) of view-height used by a single image. Default is 80.

DECLARATION

```
property ImagesZoom: double;
```

DESCRIPTION

ImagesZoom specifies the zoom (0..1) for non-central images. Default is 0.7.

DECLARATION

```
property RotateAngle: double;
```

DESCRIPTION

RotateAngle specifies the non-central images rotation in degrees (0..359). Default is 40 degrees.

Demos
viewers\manualFlow
viewers\mviewFlow

Chapter 29

TIECircularFlow

Chapter 29. IECircularFlow

```
TIECircularFlow = class(TIEAnimation);
```

DESCRIPTION

The TIECircularFlow class implements a circular animation.

Demos
viewers\manualFlow
viewers\mviewFlow

Methods and Properties

CurrentImageZoom
EllipseAngle
ImagesSizePercentage
ImagesZoom
VisibleImages

Methods and Properties

DECLARATION

```
property CurrentImageZoom: double;
```

DESCRIPTION

CurrentImageZoom specifies the zoom (0..1) for central image. The default is 1.0.

DECLARATION

```
property EllipseAngle: double;
```

DESCRIPTION

Specifies the ellipse (the curve where images are placed) angle in degrees. Default is 0 degrees.

DECLARATION

```
property ImagesSizePercentage: double;
```

DESCRIPTION

Specifies the percentage (1..100) of view size used for images. Default is 40.

DECLARATION

```
property ImagesZoom: double;
```

DESCRIPTION

Specifies the zoom (0..1) for non-central images. Default is 0.2.

DECLARATION

```
property VisibleImages: integer;
```

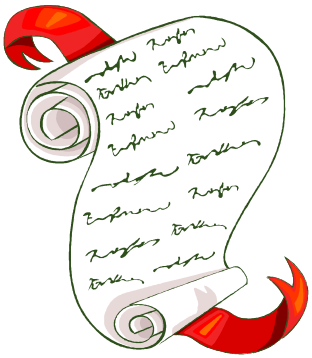
DESCRIPTION

VisibleImages specifies the number of visible images. Default is 15.

Chapter 30

TIEImageList

Chapter 30. IEImageList



DECLARATION

```
TIEImageList = class;
```

DESCRIPTION

TIEImageList is a simple in memory or file list of TIEBitmap images.

TIEImageList is one of my favorite ImageEn classes because of its versatility , ease of use and simple coding requirements. TIEImageList could easily be over looked by developers because it is a recent addition to ImageEn and because it is not a component. TIEImageList is an excellent tool to use especially with small images like glyphs, small bitmaps or icons. It can often replace a much harder to use TPageControl that contains a TImageEnView on each page which requires much more code to create the tabsheets and ImageEnView at runtime. So for a glyph or icon viewer or editor type of applications or picture viewer or editor it will be a valuable addition to your VCL toolkit.

TIEImageList is a class and is not a component. TIEImageList is a simple list of TIEBitmap images. TIEImageList contains just pointers to TIEBitmap objects but it is very similar to TImageList, except it is non-visual and can hold any image supported by ImageEnIO of any dimensions or color depth.

The TIEImageList can be set to use memory, file, or bitmap for storage. Each image in the TIEImageList can be set to different storage locations depending of the dimensions of the TIEBitmap by setting the TIEBitmap storage location.

The TIEImageList class can store TIEBitmaps in either memory and/or a file. The storage location is set by the TIEBitmap.Location property. Storage locations can be different for any image. The TIEImageList automatically sets the storage location if the image cannot be stored in memory. For example, if you set TIEBitmap.Location := ieMemory, but the actual image cannot be stored in memory, then Location for that image is automatically set to ieFile.

See Also IEBitmap.Location

Methods and Properties

AppendImageRef	Clear	Filename
FillFromDirectory	Image	ImageCount
Remove		

Methods and Properties

DECLARATION

```
function AppendImageRef(image: TIEBitmap;
filename: WideString): integer;
```

DESCRIPTION

AppendImageRef appends the specified image. This method doesn't copy the bitmap, but just takes ownership of the image object.

Parameter	Description
image	Image to append.
filename	Filename of the image.

DECLARATION

```
procedure Clear();
```

DESCRIPTION

Clear removes all images.

DECLARATION

```
property Filename[index: integer]: WideString;
```

DESCRIPTION

Filename represents filename of the specified image.

DECLARATION

```
procedure FillFromDirectory(const Directory:
WideString; Limit: integer=-1;
AllowUnknownFormats: Boolean=false;  const
ExcludeExtensions: WideString=”;
DetectFileFormat:  Boolean=false; const
FilterMask:  WideString=”);
```

DESCRIPTION

FillFromDirectory automatically loads all known images inside Directory.

Parameter	Description
Directory	Directory where to search files.
Limit	Specifies the maximum number of images to load. -1 means no limit.
AllowUnknownFormats	If false (default) loads only known and supported file formats. Otherwise tries to load all files.
ExcludeExtensions	Contains a comma separated list of file extensions to discard (i.e. 'lyr,all,iev').
DetectFileFormat	If true then the image type is detected reading the header, otherwise ImageEn looks at filename extension.
FilterMask	Contains a comma separated list of file extensions to include. Empty string means "all supported extensions".

IEImageList

DECLARATION

```
property Image[index:  integer]: TIEBitmap;
```

DESCRIPTION

Image gets/sets the specified image index.

DECLARATION

```
property ImageCount:  integer;
```

DESCRIPTION

Returns number of images in the list.
Read-only.

DECLARATION

```
procedure Remove(imageIndex: integer);
```

DESCRIPTION

Remove removes the specified image.

IEImageList

Chapter 31

Public Variables

Chapter 31. Public Variables

Public Variables are variables that control general ImageEn operations.

Unit ImageEn
Public Variables

Public Variables

DefTEMPPATH

gBlueToGrayCoef

gDefaultDPIX

gDefaultDPIY

gGreenToGrayCoef

gRedToGrayCoef

IEConvertColorFunction

IEDefDialogCenter

IEDefMinFileSize

iegAutoFragmentBitmap

iegAutoLocateOnDisk

iegColorReductionAlgorithm

iegColorReductionQuality

iegDialogsBackground

iegDefaultCoresCount

iegDefaultPreviewsZoomFilter

iegEnableCMS

iegGRIDPEN

iegMemoShortCuts

iegMINZOOMDISPLAYGRID

iegObjectsTIFFTag

iegOpSys

iegPanZoomQualityFilter

iegPreviewImageBackgroundColor

iegPreviewImageBackgroundStyle

iegUseCMYKProfile

iegUseGDIPlus

iegUseRelativeStreams

iegUseDefaultFileExists

Unit ImageEn
Public Variables

DECLARATION

```
DefTEMPPATH:  string;
```

DESCRIPTION

The DefTEMPPATH global variable specifies the directory where ImageEn stores temporary files. The default value (empty string) is equal to default system temporary directory.

Applications should set DefTEMPPATH inside 'initialization' block, before that the ImageEn components are created. For TImageEnMView component, you can change DefTEMPPATH at any time, just call after the Clear method.

DECLARATION

```
gBlueToGrayCoef: integer;
```

DESCRIPTION

Specify the blue coefficient used to convert from color to gray scale. Default values are 21, 71 and 8. The conversion formula is: gray := (Red * gRedToGrayCoef + Green * gGreenToGrayCoef + Blue * gBlueToGrayCoef) div 100;

Unit ImageEn
Public Variables

DECLARATION

```
gDefaultDPIX:  integer;
```

DESCRIPTION

gDefaultDPIX specify the default DPI X value assigned when a loaded image doesn't contain this information. Default value is 300 for both X and Y.

DECLARATION

```
gDefaultDPIY:  integer;
```

DESCRIPTION

gDefaultDPIY specify the default DPI Y value assigned when a loaded image doesn't contain this information. Default value is 300 for both X and Y.

DECLARATION

```
gGreenToGrayCoef: integer;
```

DESCRIPTION

Specify the green coefficient used to convert from color to gray scale. Default values are 21, 71 and 8. The conversion formula is:
gray := (Red * gRedToGrayCoef + Green * gGreenToGrayCoef + Blue * gBlueToGrayCoef) div 100;

Unit ImageEn
Public Variables

DECLARATION

gRedToGrayCoef: integer;

DESCRIPTION

gRedToGrayCoef specify the red coefficient used to convert from color to gray scale. Default values are 21, 71 and 8. The conversion formula is:
gray := (Red * gRedToGrayCoef + Green * gGreenToGrayCoef + Blue * gBlueToGrayCoef) div 100;

DECLARATION

IEConvertColorFunction: TIEConvertColorFunction;

DESCRIPTION

Specifies the function used to convert from a color space to another.

Declaration

```
IEDefDialogCenter:  TIEDialogCenter;
```

Description

Specify a function called when open/save dialogs needs to be centered.

Declaration

```
IEDefMinFileSize: int64;
```

Description

IEDefMinFileSize specifies the default value for MinFileSize property. MinFileSize specifies the minimum memory needed by the image to allow use of memory mapped file. If the memory needed by the image is less than MinFileSize, the image will be stored in memory (also if the Location is ieFile). If the global variable IEDefMinFileSize is not -1, it overlaps the property MinFileSize value.

Unit ImageEn
Public Variables

DECLARATION

```
iegAutoFragmentBitmap: Boolean;
```

DESCRIPTION

If iegAutoFragmentBitmap **is** true (default) TIEBitmap will try to allocate chunks of memory instead of a single buffer to contain the image.

This is useful when memory is fragmented.

DECLARATION

```
iegAutoLocateOnDisk: Boolean;
```

DESCRIPTION

If iegAutoLocateOnDisk **is** true (default) TIEBitmap will allocate the image on disk if it fails to allocate on memory.

Unit ImageEn
Public Variables

DECLARATION

```
iegColorReductionAlgorithm: Integer;
```

DESCRIPTION

Specifies the algorithm used when converting from true color (24 bit) to color mapped images. Color reduction algorithm are used, for example, when you save to a GIF.
-1 = automatically select (default)
0 = Kohonen algorithm (look also iegColorReductionQuality)
1 = Median cut

DECLARATION

```
iegColorReductionQuality: integer;
```

DESCRIPTION

When iegColorReductionAlgorithm is 0, this field specifies the quality. 0=minimum quality, 100=maximum quality.
-1 means “automatically calculated” (default)

DECLARATION

```
iegDialogsBackground: (iedbDefault, iedbPaper,
iedbMetal);
```

DESCRIPTION

Specifies the default dialogs (image processing, printing, ..) background style. This is not the image background, but actually the dialog background.

DECLARATION

```
iegDefaultCoresCount: integer;
```

DESCRIPTION

iegDefaultCoresCount specifies the total amount of processor's cores to use. -1 (default) means to get this value from operating system.

See also
IEGetCoresCount.

DECLARATION

```
iegDefaultPreviewsZoomFilter: TResampleFilter;
```

DESCRIPTION

Default zoom (resampling) filter for previews. Default is rfFastLinear.

Unit ImageEn
Public Variables

DECLARATION

```
iegEnableCMS: Boolean;
```

DESCRIPTION

Enables a Color Management System. Default CMS is provided by Windows. Default values is false: this means that no color profile is applied. Look at The Color Management System.

DECLARATION

```
iegGRIDPEN: TPen;
```

DESCRIPTION

Specifies the pen used to draw the grid.

Unit ImageEn
Public Variables

DECLARATION

```
iegMemoShortCuts: TIEMemoShortCuts;
```

DESCRIPTION

This public variable contains all key shortcuts used in Memo objects. Defaults are:

iegMemoShortCuts[iesLEFTALIGN] := ShortCut(Word('L'),[ssCtrl]);
iegMemoShortCuts[iesCENTERALIGN] := ShortCut(Word('E'),[ssCtrl]);
iegMemoShortCuts[iesRIGHTALIGN] := ShortCut(Word('R'),[ssCtrl]);
iegMemoShortCuts[iesJUSTIFIED] := ShortCut(Word('J'),[ssCtrl]);
iegMemoShortCuts[iesCOPY] := ShortCut(Word('C'),[ssCtrl]);
iegMemoShortCuts[iesCUT] := ShortCut(Word('X'),[ssCtrl]);
iegMemoShortCuts[iesPASTE] := ShortCut(Word('V'),[ssCtrl]);
iegMemoShortCuts[iesFONTSELECT] := ShortCut(Word('F'),[ssCtrl]);
iegMemoShortCuts[iesBOLD] := ShortCut(Word('B'),[ssCtrl]);
iegMemoShortCuts[iesITALIC] := ShortCut(Word('I'),[ssCtrl]);
iegMemoShortCuts[iesUNDERLINE] := ShortCut(Word('U'),[ssCtrl]);
iegMemoShortCuts[iesBACKCOLORSELECT] :=
ShortCut(Word('G'),[ssCtrl]);

Change one of above lines to set your custom short cut.

The following table specifies the default key map:

Shortcut	Function
F2	Increase font size
F1	Decrease font size
CTRL - L	Left align
CTRL - E	Center align
CTRL - R	Right align
CTRL - J	Justified
CTRL - C	Copy
CTRL - X	Cut
CTRL - V	Paste
CTRL - F	Open font dialog
CTRL - B	Bold
CTRL - I	Italic
CTRL - U	Underline
CTRL - G	Set background color (open dialog)

Unit ImageEn
Public Variables

DECLARATION

```
iegMINZOOMDISPLAYGRID: integer;
```

DESCRIPTION

Specifies the minimum value of zoom to display grid (when DisplayGrid is True). Default value is 400.

DECLARATION

```
iegObjectsTIFFTag: integer;
```

DESCRIPTION

Specifies the TIFF tag used to read/write embedded vectorial objects. The default value is 40101.

DECLARATION

```
iegOpSys: TIEOpSys;
```

DESCRIPTION

iegOpSys contains the detected operating system.

Declaration

```
iegPanZoomQualityFilter: TResampleFilter = rfNone;
```

Description

Specifies zoom filter for iettPanZoom transition effect.

Declaration

```
iegPreviewImageBackgroundColor: TColor=clBtnFace;
```

Description

Specify the background color for image preview in open/save dialogs.

Declaration

```
iegPreviewImageBackgroundStyle: TIEBackgroundStyle=iebsChessboard;
```

Description

Specify the background style for image preview in open/save dialogs.

DECLARATION

```
iegUseCMYKProfile: Boolean;
```

DESCRIPTION

When iegUseCMYKProfile **is** true ImageEn uses an internal ICC profile to convert from CMYK to RGB and vice versa.

DECLARATION

```
iegUseGDIPlus: Boolean;
```

DESCRIPTION

If iegUseGDIPlus is true ImageEn uses GDIPlus instead of GDI, when available.

DECLARATION

```
iegUseRelativeStreams: Boolean;
```

DESCRIPTION

For default ImageEn uses absolute streams. This means that the image must be contained from position zero of the stream. Setting iegUseRelativeStreams=true, you can put the image in any offset of the stream. Default is False.

DECLARATION

```
iegUseDefaultFileExists: Boolean;
```

DESCRIPTION

When true (default is false since version 3.0.3) ImageEn uses FileExists function. Otherwise uses an different way which doesn't need "listing" Windows rights.

Chapter 32

HYIEDefs

Chapter 32. HYIEDefs

ImageEn provides a number of definitions in the HYIEDefs.

DefTEMPPATH

gBlueToGrayCoef

gDefaultDPIX

gDefaultDPIY

gGreenToGrayCoef

gRedToGrayCoef

IEConvertColorFunction

IEDefDialogCenter

IEDefMinFileSize

iegAutoFragmentBitmap

Unit HYIEDefs

Definitions

iegAutoLocateOnDisk

iegColorReductionAlgorithm

iegColorReductionQuality

iegDialogsBackground

iegDefaultCoresCount

iegDefaultPreviewsZoomFilter

iegEnableCMS

iegGRIDPEN

iegMemoShortCuts

iegMINZOOMDISPLAYGRID

iegObjectsTIFFTag

iegOpSys

iegPanZoomQualityFilter

iegPreviewImageBackgroundColor

iegPreviewImageBackgroundStyle

iegUseCMYKProfile

iegUseGDIPlus

iegUseRelativeStreams

iegUseDefaultFileExists

IntegerArray

pboolean

PBYTEROW

PBYTEROWS

PCIELAB

PCIELABROW

PCMYK

PCMYKROW

PDoubleArray

PDWordArray

PIEPointArray

PIERGBAPalette

PIERGBPalette

PIESmallIntArray

pIntegerArray

PPointArray

PPointerArray

PPRGBArray

PProgressRec

PRGB

PRGB32ROW

PRGB48

PRGB48ROW

PRGBA

PRGBArray

PRGBROW

PSingleArray

Unit HYIEDefs

Definitions

PYCBCR

RGB32ROW

RGBROW

TBYTEROW

TBYTEROWS

TCIELab

TCIELABROW

TCMYK

TCMYK16

TCMYKROW

TDoubleArray

TDWordArray

TIEJob

TIEJobEvent

TIEPoint

TIEPointArray

TIEProgressEvent

TIEQuadCoords

TIERectangle

TIERGBAPalette

TIERGBPalette

TIESmallIntArray

TMsgLanguage

TMsgLanguageWords

TPointArray

TPointerArray

TProgressRec

TResampleFilter

TRGB

TRGB48

TRGB48ROW

TRGBA

TSingleArray

TYCbCr

Unit HYIEDefs

Definitions

DECLARATION

```
IntegerArray = array[0..MaxInt div 16] of
integer;
```

DECLARATION

```
pboolean = ^boolean;
```

DECLARATION

```
PBYTEROW = ^TBYTEROW;
```

DECLARATION

```
PBYTEROWS = ^TBYTEROWS;
```

DECLARATION

```
PCIELAB = ^TCIELAB;
```

DECLARATION

```
PCIELABROW = ^TCIELABROW;
```

DECLARATION

```
PCMYK = ^TCMYK;
```

Unit HYIEDefs

Definitions

Declaration

```
PCMYKROW = ^TCMYKROW;
```

Declaration

```
PDoubleArray = ^TDoubleArray;
```

Declaration

```
PDWordArray = ^TDWordArray;
```

Declaration

```
PIEPointArray = ^TIEPointArray;
```

Declaration

```
PIERGBAPalette = ^TIERGBAPalette;
```

Declaration

```
PIERGBPalette = ^TIERGBPalette;
```

Unit HYIEDefs

Definitions

DECLARATION

```
PIESmallIntArray = ^TIESmallIntArray;
```

DECLARATION

```
pIntegerArray = ^IntegerArray;
```

DECLARATION

```
PPointArray = ^TPointArray;
```

DECLARATION

```
PPointerArray = ^TPointerArray;
```

DECLARATION

```
type PPRGBArray = ^PRGBArray;
```

DECLARATION

```
PProgressRec = ^TProgressRec;
```

DECLARATION

```
type PRGB = ^TRGB;
```

HYIEDefs

Unit HYIEDefs

Definitions

DECLARATION

```
type PRGB32ROW = ^RGB32ROW;
```

DECLARATION

```
PRGB48 = ^TRGB48;
```

DECLARATION

```
PRGB48ROW = ^TRGB48ROW;
```

DECLARATION

```
PRGBA = ^TRGBA;
```

DECLARATION

```
type PRGBArray = array[0 . . Maxint div 16] of
pRGB;
```

DECLARATION

```
type PRGBROW = ^RGBROW;
```

HYIEDefs

Unit HYIEDefs

Definitions

DECLARATION

```
PSingleArray = ^TSingleArray;
```

DECLARATION

```
PYCBCR = ^TYCBCR;
```

DECLARATION

```
type RGB32ROW = array[0 . . Maxint div 16] of
TRGBA;
```

DECLARATION

```
type RGBROW = array[0 . . Maxint div 16] of TRGB;
```

DECLARATION

```
TBYTEROW = array[0..Maxint div 16] of byte;
```

DECLARATION

```
TBYTEROWS = array[0..Maxint div 16] of PBYTEROW;
```

HYIEDefs

Unit HYIEDefs

Definitions

Declaration

```
TCIELab
```

Declaration

```
T3dPoint = record
x, y, z: double;
end;
```

Declaration

```
TCMYK = packed record
c: byte;
m: byte;
y: byte;
k: byte;
end;
```

Declaration

```
TCMYK16 = packed record
c: word;
m: word;
y: word;
k: word;
end;
```

Unit HYIEDefs
Definitions

DECLARATION

```
TCMYKROW = array[0..Maxint div 16] of TCMYK;
```

DECLARATION

```
TDoubleArray = array[0..maxint div 16] of double;
```

DECLARATION

```
TDWordArray = array[0..maxint div 16] of dword;
```

DECLARATION

```
TIEJob = (iejNOTHING, // no job / end of work
iejGENERALPROCESSING, // generic processing
iejVIDEOCAP_CONNECTING // Video capture -
connecting...
);
```

Unit HYIEDefs
Definitions

Declaration

```
type TIEJobEvent = procedure(Sender: TObject;
job: TIEJob; per: integer) of object;
```

Description

A TIEJobEvent is called during ImageEn processing.

Job is the job type in progress. Per is percentage of performed work related to job.

The job parameter can be one of these values:

iejNOTHING : No job / end of work
iejGENERALPROCESSING : generic job
iejVIDEOCAP_CONNECTING : video capture - connecting to video capture card

Declaration

```
TIEPoint = record
X: Longint;
Y: Longint;
end;
```

Unit HYIEDefs

Definitions

DECLARATION

```
TIEPointArray = array[0..Maxint div 16] of
TIEPoint;
```

DECLARATION

```
type TIEProgressEvent = procedure(Sender:
TObject; per: integer) of object;
```

DESCRIPTION

Per is a value from 0 to 100 that indicates the percentage progress made.

DECLARATION

```
TIEQuadCoords = record
x0, y0: integer;
x1, y1: integer;
x2, y2: integer;
x3, y3: integer;
end;
```

Unit HYIEDefs

Definitions

Declaration

```
TIERectangle = packed record
x, y: integer;
width, height: integer;
end;
```

Description

Describes a rectangle in terms of top-left, top-right coordinates and sizes.

Declaration

```
TIERGBAPalette = array[0..maxint div 16] of
TRGBA;
```

Declaration

```
TIERGBPalette = array[0..maxint div 16] of TRGB;
```

Declaration

```
TIESmallIntArray = array[0..maxint div 16] of
SmallInt;
```

Unit HYIEDefs
Definitions

DECLARATION

```
Type TMsgLanguage=(msSystem, msEnglish,
msItalian, msGerman, msSpanish, msFrench,
msPortuguese, msGreek, msRussian, msDutch,
msSwedish, msPolish, msJapanese, msCzech,
msFinnish, msFarsi, msChinese,
msChineseTraditional, msDanish, msTurkish,
msKorean, msHungarian, msUser);
```

DECLARATION

```
TMsgLanguageWords = (IEMSG_PREVIEW, IEMSG_SOURCE,
IEMSG_RESULT, IEMSG_OK, IEMSG_CANCEL,
IEMSG_LOCKPREVIEW, IEMSG_COPYRESULTTOSOURCE,
IEMSG_CONTRAST, IEMSG_BRIGHTNESS,
IEMSG_HUE, IEMSG_SATURATION, IEMSG_VALUE,
IEMSG_BASECOLOR, IEMSG_NEWCOLOR,
IEMSG_LUMINOSITY, IEMSG_RED, IEMSG_GREEN,
IEMSG_BLUE, IEMSG_FILTERVALUES, IEMSG_PRESETS,
IEMSG_DIVISOR, IEMSG_LOAD, IEMSG_SAVE,
IEMSG_EQUALIZATION, IEMSG_THRESHOLD,
IEMSG_EQUALIZE,
IEMSG_HISTOGRAM, IEMSG_GRAY, IEMSG_LIGHT,
IEMSG_LEFT, IEMSG_TOP, IEMSG_WIDTH, IEMSG_HEIGHT,
IEMSG_COLOR, IEMSG_SOURCEIMAGEQUANTITY,
IEMSG_LENS, IEMSG_REFRACTION,
IEMSG_PARAMETERSPREVIEW,
IEMSG_QUALITY, IEMSG_DCTMETHOD,
IEMSG_SMOOTHINGFACTOR, IEMSG_GRAYSCALE,
IEMSG_OPTIMALHUFFMAN,
```

```
IEMSG_ORIGINALSIZE, IEMSG_COMPRESSEDSIZE,
IEMSG_ADVANCED, IEMSG_PROGRESSIVE,
IEMSG_COMPRESSION,
IEMSG_IMAGEINDEX, IEMSG_PHOTOMETRIC,
IEMSG_SCANNDEDDOCUMENTINFO, IEMSG_NAME,
IEMSG_DESCRIPTION,
IEMSG_PAGENAME, IEMSG_PAGENUMBER, IEMSG_OF,
IEMSG_HORIZPOSITIONINCH, IEMSG_VERTPOSITIONINCH,
IEMSG_COLORS, IEMSG_TRANSPARENT,
IEMSG_TRANSPARENTCOLOR, IEMSG_INTERLACED,
IEMSG_BACKGROUND,
IEMSG_HORIZPOSITION,
IEMSG_VERTPOSITION,IEMSG_DELAYTIME, IEMSG_FILTER,
IEMSG_WAVE,
IEMSG_AMPLITUDE, IEMSG_WAVELENGTH, IEMSG_PHASE,
IEMSG_REFLECTIVE, IEMSG_USERFILTER,
IEMSG_MORPHFILTER, IEMSG_WINDOWSIZE,
IEMSG_MAXIMUM, IEMSG_MINIMUM, IEMSG_OPEN,
IEMSG_CLOSE, IEMSG_ROTATE, IEMSG_FLIP,
IEMSG_FLIPHOR, IEMSG_FLIPVER,
IEMSG_FREQUENCYDOMAINIMAGE,
IEMSG_SELECTTHEREGIONTOCLEAR, IEMSG_CLEAR,
IEMSG_RESET,
IEMSG_ANIMATE, IEMSG_LOADFILTER,
IEMSG_SAVEFILTER, IEMSG_BUMPMAP, IEMSG_PRINT,
IEMSG_MARGINS, IEMSG_INCHES,
IEMSG_RIGHT, IEMSG_BOTTOM, IEMSG_POSITION,
IEMSG_SIZE, IEMSG_NORMAL, IEMSG_FITTOPAGE,
IEMSG_STRETCHTOPAGE,
IEMSG_SPECIFIEDSIZE, IEMSG_GAMMACORRECTION,
IEMSG_VALUE2, IEMSG_PRINTSETUP, IEMSG_LEFTMARGIN,
```

```
IEMSG_TOPMARGIN,
IEMSG_RIGHTMARGIN, IEMSG_BOTTOMMARGIN,
IEMSG_LOCATIONSIZE, IEMSG_TOPLEFT,
IEMSG_TOPCENTER, IEMSG_TOPRIGHT,
IEMSG_CENTERLEFT, IEMSG_CENTER,
IEMSG_CENTERRIGHT, IEMSG_BOTTOMLEFT,
IEMSG_BOTTOMCENTER, IEMSG_BOTTOMRIGHT,
IEMSG_CLOSE2, IEMSG_APPLY, IEMSG_MEASUREUNITS,
IEMSG_UNITS, IEMSG_RATE,
IEMSG_ALLGRAPHICS, IEMSG_VIDEOFORWINDOWS,
IEMSG_FILE, IEMSG_MEM, IEMSG_LOCKPREVIEWHINT,
IEMSG_PRINTALL, IEMSG_PRINTSELECTED,
IEMSG_COMMONGRAPHICFILES, IEMSG_ALLFILES,
IEMSG_HSV, IEMSG_HSL, IEMSG_RGB, IEMSG_FFT,
IEMSG_SHARPEN, IEMSG_CHANNELS,
IEMSG_PIXEL, IEMSG_FRAMES, IEMSG_BIT, IEMSG_BYTE,
IEMSG_DPI, IEMSG_KB, IEMSG_FIT, IEMSG_OTHER,
IEMSG_COLUMNS, IEMSG_ROWS, IEMSG_STYLE,
IEMSG_SPACING
);
```

DECLARATION

```
TPointArray = array[0..Maxint div 16] of
Windows.TPoint;
```

DECLARATION

```
TPointerArray = array[0..maxint div 16] of
pointer;
```

Unit HYIEDefs

Definitions

Declaration

```
// helper structure for OnProgress
TProgressRec = record
fOnProgress: TIEProgressEvent;
Sender: TObject;
val: integer; // counter (per = trunc(per1*val))
tot: integer;
per1: double; // (100/maxval)
per2: double;
Aborting: pboolean;
end;
```

Declaration

```
TResampleFilter = (rfNone, rfTriangle, rfHermite,
rfBell, rfBSpline, rfLanczos3, rfMitchell,
rfNearest, rfLinear, rfFastLinear, rfBilinear,
rfBicubic, rfProjectBW, rfProjectWB);
```

Description

If you need the best quality we suggest: rfHermite, rfBell, rfBSpline, rfLanczos3, rfMitchell, rfNearest, rfBilinear, rfBicubic

If speed is required then we suggest: rfTriangle, rfLinear, rfFastLinear

For projects (white on black or black on white) we suggest: rfProjectBW and rfProjectWB

Unit HYIEDefs

Definitions

Declaration

```
TRGB = packed record
b: byte;
g: byte;
r: byte;
end;
```

Declaration

```
TRGB48 = packed record
r: word;
g: word;
b: word;
end;
```

Declaration

```
TRGB48ROW = array[0..Maxint div 16] of TRGB48;
```

Declaration

```
TRGBA = packed record
b: byte;
g: byte;
r: byte;
a: byte;
end;
```

Unit HYIEDefs

Definitions

DECLARATION

```
TSingleArray = array[0..maxint div 16] of single;
```

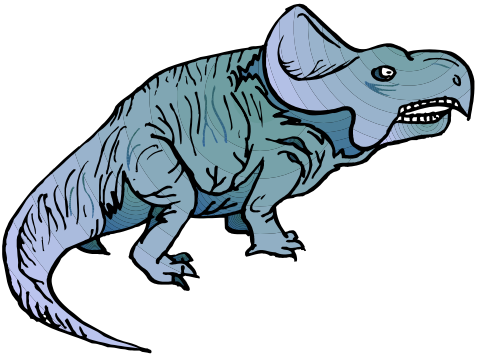
DECLARATION

```
TYCbCr = packed record
y: byte;
Cb: byte;
Cr: byte;
end;
```

Chapter 33

Helper Functions

Chapter 33. Helper Functions



ImageEn provides a number of helper functions to assist you when developing with the ImageEn Library.

Helper Functions

Generic File Formats

FindFileFormat
FindStreamFormat
IEAddExtIOPlugin
IEExtToFileFormat
IEFindNumberWithKnownFormat
IEGetFileFramesCount
IEIsInternalFormat
IsKnownFormat

GIF File Format

CheckAniGIF
DeleteGifIm
EnumGifIm
IEOptimizeGIF

TIFF File Format

DeleteTIFFImGroup
DeleteTIFFIm
EnumTIFFIm
EnumTIFFStream
ExtractTIFFImageFile
ExtractTIFFImageStream
InsertTIFFImageFile
InsertTIFFImageStream

ICO File Format

EnumICOIm
IEWriteICOImages

Unit HYIEUtils
Helper Functions

DCX File Format

DeleteDCXIm EnumDCXIm

JPEG

JpegLosslessTransform JpegLosslessTransform2

JpegLosslessTransformStream IECalcJpegFileQuality

IECalcJpegStreamQuality

Register ImageEn File Formats In VCL

IERegisterFormats IEUnregisterFormats

AVI File Format

IEAVISelectCodec IEAVIGetCodecs

IEAVIGetCodecsDescription

Layers

IELayersMerge

Utilities

IEAlphaToOpacity IEGetCoresCount IEOpacityToAlpha

Generic File Formats

DECLARATION

```
function FindFileFormat(const FileName:
WideString; VerifyExtension:
Boolean):TIOFileType;
```

DESCRIPTION

Returns file format of the specified file. FindFileFormat reads the file and try to recognize file format from file header.

If VerifyExtension is False, the FindFileFormat function doesn't examine the file extension.

FineFileFormat uses FindStreamFormat to detect image format from file header.

DECLARATION

```
function FindStreamFormat(Stream:
TStream):TIOFileType;
```

DESCRIPTION

FindStreamFormat returns the format of the specified file. FindStreamFormat reads the stream and tries to recognize the file's format from the file's header.

This function can recognize one of following file formats:

Return value	File format
ioGIF	GIF
ioBMP	BMP
ioPCX	PCX
ioTIFF	TIFF
ioPNG	PNG
ioDICOM	DICOM
ioICO	ICO
ioCUR	CUR
ioTGA	TGA
ioPXM	PXM

Return value	File format
ioJPEG	JPEG
ioJP2, ioJ2K	JPEG2000
ioAVI	AVI
ioDCX	DCX
ioWMF	WMF
ioEMF	EMF
ioPSD	PSD
ioIEV	IEV (ImageEn vectorial objects)
ioLYR	LYR (ImageEn layers)
ioALL	ALL (ImageEn vectorial objects and layers)
ioRAW	RAW (when dcraw plugin is not installed)
ioHDP	Microsoft HD Photo
ioDLLPLUGINS + offset	External plugins (ie dcraw)
ioUSER + offset	User registered file formats

Unit HYIEUtils
Helper Functions

Declaration

```
function IEAddExtIOPlugin(const FileName:
string):integer;
```

Description

This function adds new file formats using an external dynamic library (dll), named input/output plugin. We suggest to download examples at http://www.hi-components.com/ndownloads_plgins.asp here you can find plugins to load and save JBIG, the last release of DCRAW (to load camera RAW) and other useful plugins. All of them are inclusive of source code, so you can know how write new ones.

IEAddExtIOPlugin returns the file type that you can use, for example, in LoadFromStreamFormat.

Declaration

```
function IEExtToFileFormat(ex: string):
TIOFileType;
```

Description

Converts file extension to TIOFileType type.

Declaration

```
function IEFindNumberWithKnownFormat( const
Directory: WideString ): integer;
```

Description

This function returns the number of images recognized by ImageEn in specified directory.

Declaration

```
function IEGetFileFramesCount(const FileName:
WideString):integer;
```

Description

This is a generic way to get number of frames in a multipage file (GIF, TIFF, AVI, MPEG...) or a single page (JPEG...).

Declaration

```
function IEIsInternalFormat(ex: string): Boolean;
```

Description

Returns true if the specified extension is recognized as internal file format.

DECLARATION

```
function IsKnownFormat(const FileName:
WideString): Boolean;
```

DESCRIPTION

Returns true if the specified filename is a known file format.

GIF File Format

DECLARATION

```
function CheckAniGIF(const FileName:
WideString):Boolean;
```

DESCRIPTION

Returns True if the GIF image FileName has the animated flag set ("NETSCAPE2.0" string). If the file doesn't exist or if the image hasn't set the flag, it returns False.

Declaration

```
function DeleteGIFIm(const FileName: WideString;
idx: integer):integer;
```

Description

Remove the idx (0 is first image) image from FileName file. DeleteGIFIm returns the remaining number of frames that FileName file contains. If the FileName doesn't exists or doesn't contains images return 0.

Declaration

```
function EnumGIFIm(const FileName: WideString):
integer;
```

Description

EnumGIFIm returns the number of frames (images) that FileName file contains. A GIF could contain more images also if it is not marked as animated. If the FileName doesn't exist or doesn't contain images, it returns a value of 0.

DECLARATION

```
procedure IEOptimizeGIF(const
InputFile,OutputFile: WideString);
```

DESCRIPTION

This procedure optimizes a GIF animations (or multipage) discovering differences among frames and saving in the new GIF only the differences.

TIFF File Format

DECLARATION

```
function DeleteTIFFImGroup(const FileName:
WideString; Indexes: array of integer):integer;
```

DESCRIPTION

DeleteTIFFImGroup removes the specified group of pages from the FileName file. DeleteTIFFImGroup returns the remaining number of frames that FileName file contains.

Indexes are an array of page indexes to remove. If the FileName doesn't exists or doesn't contains images return 0.

Declaration

```
function DeleteTIFFIm(const FileName: WideString;
idx: integer):integer;
```

Description

Remove the idx (0 is first image) image from FileName file. DeleteTIFFIm returns the remaining number of frames that FileName file contains. If the FileName doesn't exists or doesn't contains images return 0.

Declaration

```
function EnumTIFFIm(const FileName:
WideString):integer;
```

Description

EnumTIFFIm returns the number of frames (images) that FileName file contains. If the FileName doesn't exist or doesn't contain images, it returns a value of 0.

Declaration

```
function EnumTIFFStream(Stream: TStream):integer;
```

Description

EnumTIFFStream returns the number of frames (images) that the TIFF Stream contains.

Declaration

```
procedure ExtractTIFFImageFile(const
SourceFileName, OutFileName: WideString; idx:
integer);
```

Description

ExtractTIFFImageFile extracts the page idx (starting at 0) from SourceFileName and saves it to OutFileName. It doesn't remove the page from source file; also it doesn't decompress the image resulting in a very quick process.

Declaration

```
procedure ExtractTIFFImageStream(SourceStream,
OutStream: TStream; idx: integer);
```

Description

ExtractTIFFImageStream extracts the page idx (starting at 0) from SourceStream and saves it to OutStream. It doesn't remove the page from source file; also it doesn't decompress the image resulting in a very quick process.

Declaration

```
procedure InsertTIFFImageFile(const
SourceFileName, InsertingFileName, OutFileName:
WideString; idx: integer);
```

Description

InsertTIFFImageFile inserts the file InsertingFileName in SourceFileName saving result in OutFileName.

Idx is the page where to insert the file. It is a fast operation because both sources and destination images are uncompressed.

The two TIFFs must have the same byte order. You can check it reading image parameters and checking TIFF_ByteOrder property.

DECLARATION

```
procedure
InsertTIFFImageStream(SourceStream,InsertingStrea
m,OutStream: TStream; idx: integer);
```

DESCRIPTION

InsertTIFFImageStream inserts the TIFF stream InsertingStream in SourceStream saving result in OutStream. Idx is the page where to insert the file.

The two TIFFs must have the same byte order. You can check it reading image parameters and checking TIFF_ByteOrder property.

It is a fast operation because both sources and destination images are uncompressed.

ICO File Format

DECLARATION

```
function EnumICOIm(const FileName:
WideString):integer;
```

DESCRIPTION

EnumICOIm returns the number of images inside an ICO file.

DECLARATION

```
procedure IEWriteICOImages(const fileName:
WideString; images: array of TObject);
```

DESCRIPTION

IEWriteICOImages is an alternate mode to save ICO files (instead SaveToFile or SaveToFileICO). It allows specifying the origin of each frame of ICO to be saved. IEWriteICOImages doesn't look at ICO_Sizes and ICO_BitCount, but just to TImageEnView.IO.Params.BitsPerSample and TImageEnView.IO.Params.SamplesPerPixel.

DCX File Format

DECLARATION

```
procedure DeleteDCXIm(const FileName: WideString;
idx: integer);
```

DESCRIPTION

Remove the idx (0 is first image) image from FileName file.

If the FileName doesn't exists or doesn't contains images return 0.

Declaration

```
function EnumDCXIm(const FileName:
WideString):integer;
```

Description

EnumDCXIm returns the number of frames (images) that FileName file contains.

If the FileName doesn't exist or doesn't contain images, it returns a value of 0.

JPEG File Format

DECLARATION

```
function JpegLosslessTransform(const SourceFile,
DestFile: WideString; Transform:
TIEJpegTransform; GrayScale: Boolean;
CopyMarkers: TIEJpegCopyMarkers; CutRect: Trect,
UpdateEXIF: Boolean=false):Boolean;
function JpegLosslessTransform(const SourceFile,
DestFile: WideString; Transform:
TIEJpegTransform): Boolean; overload;
```

DESCRIPTION

JpegLosslessTransform performs various useful transformations of Jpeg files. JpegLosslessTransform works by rearranging the compressed data, without ever fully decoding the image. Therefore, its transformations are lossless: there is no image degradation at all.

The resulting image will contain at least the CutRect region (constrained to the image size), and may be larger to round up to the nearest DCT block boundary (or multiple thereof, depending on the sampling factors). If the image ends in a partial DCT block that part of the image will be lost, even though it would be possible to include it in the image.

SourceFile and DestFile are file path of source and destination Jpeg files. Transform specifies the required transformation. GrayScale = True force grayscale output. CopyMarkes specifies how to copy comments and markers (as IPTC info). CutRect specifies the rectangle to save when jtCut is specified.

UpdateEXIF: when true ImageEn updates orientation and thumbnail of EXIF tags. JpegLosslessTransform returns false if it fails.

Declaration

```
function JpegLosslessTransform2(const FileName:
WideString; Transform: TIEJpegTransform;
GrayScale: Boolean; CopyMarkers:
TIEJpegCopyMarkers; CutRect: TRect; UpdateEXIF:
Boolean): Boolean;
```

Description

This function is like JpegLosslessTransform, but gets only one FileName which is both input and output.

DECLARATION

```
function
JpegLosslessTransformStream(SourceStream,
DestStream: TStream; Transform: TIEJpegTransform;
GrayScale: Boolean; CopyMarkers:
TIEJpegCopyMarkers; CutRect: Trect; UpdateEXIF:
Boolean=false): Boolean;
```

DESCRIPTION

JpegLosslessTransformStream performs various useful transformations of Jpeg files. JpegLosslessTransform works by rearranging the compressed data, without ever fully decoding the image. Therefore, its transformations are lossless: there is no image degradation at all.

The resulting image will contain at least the CutRect region (constrained to the image size), and may be larger to round up to the nearest DCT block boundary (or multiple thereof, depending on the sampling factors). If the image ends in a partial DCT block that part of the image will be lost, even though it would be possible to include it in the image.

SourceStream and DestStream are source and destination TStream objects that contains jpeg stream. Transform specifies the required transformation.

GrayScale = True forces Grayscale output.

CopyMarkes specifies how to copy comments and markers (as IPTC info). CutRect specifies the rectangle to save when jtCut is specified.

UpdateEXIF: when true ImageEn updates orientation and thumbnail of EXIF tags.

JpegLosslessTransform returns false if it fails.

DECLARATION

```
function IECalcJpegFileQuality(const FileName: WideString):integer;
```

DESCRIPTION

This function estimates the quality of a Jpeg file.
Returned value can be used with the JPEG_Quality property.

DECLARATION

```
function IECalcJpegStreamQuality(Stream: TStream): integer;
```

DESCRIPTION

This function estimates the quality of a Jpeg file.
Returned value can be used with the JPEG_Quality property.

Register ImageEn file formats in VCL

DECLARATION

```
procedure IERegisterFormats;
```

DESCRIPTION

Register/unregister following classes inside the VCL class framework. This allows use of standard VCL open/save dialogs and TPicture objects with ImageEn file formats.

TIETiffImage, TIEGifImage, TIEJpegImage, TIEPCXImage, TIEBMPImage, TIEICOImage, TIEPNGImage, TIETGAImage, TIEPXMImage, TIEJP2Image, TIEJ2Kimage.

Each of above classes inherits from TIEGraphicBase. To get input/ouput parameters use “IO” property: it is the same of ImageEnIO.Params property.

DECLARATION

```
procedure IEUnregisterFormats;
```

DESCRIPTION

IEUnregisterFormats unregisters ImageEn file formats from the VCL.

AVI File Format

DECLARATION

```
function IEAVISelectCodec: AnsiString;
```

DESCRIPTION

This function shows Windows codec selection dialog and return the (fourcc) codec string, that you can pass for example to CreateAVIFile.

DECLARATION

```
function IEAVIGetCodecs: TStringList;
```

DESCRIPTION

This function returns a list of Windows codec as (fourcc) codec strings, that you can pass for example to CreateAVIFile. To obtain codecs description use IEAVIGetCodecsDescription.

DECLARATION

```
function IEAVIGetCodecsDescription: TStringList;
```

DESCRIPTION

This function returns a list of Windows codec descriptions. To obtain actual codecs use IEAVIGetCodecs.

Layers

DECLARATION

```
procedure IELayersMerge(Layer0, Layer1,
LayerMask: TIELayer; Destination: TIEBitmap;
ZoomFilter: TResampleFilter; Background: TColor);
```

DESCRIPTION

IELayersMerge merges two layers to the destination bitmap. ZoomFilter specifies the filter to use when layers needs to be stretched. Background is the color used to fill empty regions. LayerMask is the upper layer's mask. This can be 'nil' if no layer mask exists.

Utilities

DECLARATION

```
function IEAlphaToOpacity(Alpha: integer
):integer;
```

DESCRIPTION

AlphaToOpacity returns an integer of the alpha channel value. Opacity is generally referred to as percent. (Alpha = 255 then Opacity = 100%).

DECLARATION

```
function IEGetCoresCount(): integer;
```

DESCRIPTION

IEGetCoresCount returns the total amount of processor's cores present on the system. Some Windows operating system versions cannot communicate this information, so it is defaulted to iegDefaultCoresCount public field.

Declaration

```
function IEOpacityToAlpha(Opacity: integer): integer;
```

Description

OpacityToAlpha returns an integer of an opacity value. (Opacity = 100% then Alpha = 255)

Chapter 34

Color Management System

Chapter 34. Color Management System (CMS)

ImageEn includes a Color Management System (CMS) which if enabled allows to render the image with the original colors. To enable the CMS you have to write: iegEnableCMS := True;

IegEnableCMS public variable is defined in the unit ImageEnIO.

If you want to use LCMS instead of Windows CMS you need to recompile ImageEn (with source code). Make sure that following line inside ie.inc is enabled: (*$define IEINCLUDECMS*)

If the CMS is enabled the loading of images with a color profile will be slower than if the CMS is disabled.

Unit HYIEUtils

The Color Management System

If you don't define IEINCLUDECMS or if you have only compiled version of ImageEn, only Windows CMS will be used. Note that Windows CMS is quicker than LCMS.
See also TIEICC

Unit HYIEUtils
The Color Management System

ImageEn includes parts of LittleCMS by Marti Maria.

In digital imaging systems, **color management** is the controlled conversion between the color representations of various devices, such as image scanners, digital cameras, monitors, TV screens, film printers, computer printers, offset presses, and corresponding media.

The primary goal of color management is to obtain a good match across color devices; for example, the colors of one frame of a video should appear the same on a computer LCD monitor, on a plasma TV screen, and as a printed poster. Color management helps to achieve the same appearance on all of these devices, provided the devices are capable of delivering the needed color intensities.

Parts of this technology are implemented in the operating system (OS), helper libraries, the application, and devices. A cross-platform view of color management is the use of an ICC-compatible color management system. The International Color Consortium (ICC) is an industry consortium that has defined:

- An open standard for a *Color Matching Module* (CMM) at the OS level
- color profiles for:
    - Devices, including device link-profiles that represent a complete color transformation from source device to target device
    - *Working spaces*, the color spaces in which color data is meant to be manipulated

The **International Color Consortium** was formed in 1993 by eight industry vendors in order to create an open, vendor-neutral color management system which would function transparently across all operating systems and software packages.

Unit HYIEUtils

The Color Management System

The ICC specification, currently on version 4.3[1] allows for matching of color when moved between applications and operating systems, from the point of creation to the final output, whether display or print.

LittleCMS or **LCMS** is an open source color management system, released as a software library for use in other programs which will allow the use of International Color Consortium profiles. It is licensed under the MIT License Agreement.

Little CMS intends to be a small-footprint color management engine, with special focus on accuracy and performance. It uses the International Color Consortium standard (ICC), which is the modern standard when regarding to color management. The ICC specification is widely used and is referred to in many International and other de-facto standards. It was approved as an International Standard, ISO 15076-1, in 2005

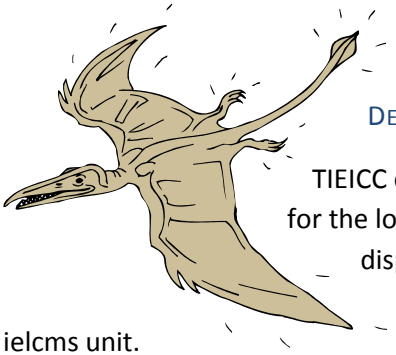
Copyright notice of lcms:

LittleCMS

Chapter 35

TIECC

Chapter 35. IEICC Color Profile



DESCRIPTION

TIEICC class contains a color profile. It is used for the loaded image ICC profile and for the display ICC profile (which default is sRGB).

Note: several constants are defined in ielcms unit.

Methods and Properties

Apply	Apply2	Assign_LabProfile
Assign_LabProfileD50	Assign_LabProfileFromTemp	Assign_sRGBProfile
Assign_XYZProfile	Assign	Clear
ConvertBitmap	Copyright	Description
FreeTransform	IsApplied	IsValid
LoadFromBuffer	LoadFromFile	LoadFromStream

SaveToStream Transform

Methods and Properties

DECLARATION

```
function Apply(SourceBitmap: TIEBitmap;
SourceFormat: integer; DestinationBitmap:
TIEBitmap; DestinationFormat: integer;
DestinationProfile: TIEICC; Intent: integer;
Flags: integer): Boolean;
```

DESCRIPTION

Apply transforms a bitmap from current profile to the destination profile. Look at Transform for parameters description. You have to call FreeTransform after the whole image has been transformed. Apply returns false if it cannot perform the transformation.

Also See

TIEICC Supported Color Formats

TIEICC Supported Color Formats

TYPE_GRAY_8
TYPE_GRAY_8_REV
TYPE_GRAY_16
TYPE_GRAY_16_REV
TYPE_GRAY_16_SE
TYPE_GRAYA_8
TYPE_GRAYA_16
TYPE_GRAYA_16_SE
TYPE_GRAYA_8_PLANAR
TYPE_GRAYA_16_PLANAR
TYPE_RGB_8
TYPE_RGB_8_PLANAR
TYPE_BGR_8
TYPE_BGR_8_PLANAR
TYPE_RGB_16
TYPE_RGB_16_PLANAR
TYPE_RGB_16_SE
TYPE_BGR_16
TYPE_BGR_16_PLANAR
TYPE_BGR_16_SE
TYPE_RGBA_8
TYPE_RGBA_8_PLANAR
TYPE_RGBA_16

TIEICC

TYPE_RGBA_16_PLANAR
TYPE_RGBA_16_SE
TYPE_ARGB_8
TYPE_ARGB_16
TYPE_ABGR_8
TYPE_ABGR_16
TYPE_ABGR_16_PLANAR
TYPE_ABGR_16_SE
TYPE_BGRA_8
TYPE_BGRA_16
TYPE_BGRA_16_SE
TYPE_CMY_8
TYPE_CMY_8_PLANAR
TYPE_CMY_16
TYPE_CMY_16_PLANAR
TYPE_CMY_16_SE
TYPE_CMYK_8
TYPE_CMYK_8_REV
TYPE_YUVK_8
TYPE_CMYK_8_PLANAR
TYPE_CMYK_16
TYPE_CMYK_16_REV
TYPE_YUVK_16
TYPE_CMYK_16_PLANAR
TYPE_CMYK_16_SE
TYPE_KYMC_8
TYPE_KYMC_16
TYPE_KYMC_16_SE

TYPE_KCMY_8
TYPE_KCMY_8_REV
TYPE_KCMY_16
TYPE_KCMY_16_REV
TYPE_KCMY_16_SE
TYPE_CMYK5_8
TYPE_CMYK5_16
TYPE_CMYK5_16_SE
TYPE_KYMC5_8
TYPE_KYMC5_16
TYPE_KYMC5_16_SE
TYPE_CMYKcm_8
TYPE_CMYKcm_8_PLANAR
TYPE_CMYKcm_16
TYPE_CMYKcm_16_PLANAR
TYPE_CMYKcm_16_SE
TYPE_CMYK7_8
TYPE_CMYK7_16
TYPE_CMYK7_16_SE
TYPE_KYMC7_8
TYPE_KYMC7_16
TYPE_KYMC7_16_SE
TYPE_CMYK8_8
TYPE_CMYK8_16
TYPE_CMYK8_16_SE
TYPE_KYMC8_8
TYPE_KYMC8_16
TYPE_KYMC8_16_SE
TYPE_CMYK9_8

TIEICC

TYPE_CMYK9_16
TYPE_CMYK9_16_SE
TYPE_KYMC9_8
TYPE_KYMC9_16
TYPE_KYMC9_16_SE
TYPE_CMYK10_8
TYPE_CMYK10_16
TYPE_CMYK10_16_SE
TYPE_KYMC10_8
TYPE_KYMC10_16
TYPE_KYMC10_16_SE
TYPE_CMYK11_8
TYPE_CMYK11_16
TYPE_CMYK11_16_SE
TYPE_KYMC11_8
TYPE_KYMC11_16
TYPE_KYMC11_16_SE
TYPE_CMYK12_8
TYPE_CMYK12_16
TYPE_CMYK12_16_SE
TYPE_KYMC12_8
TYPE_KYMC12_16
TYPE_KYMC12_16_SE
TYPE_XYZ_16
TYPE_Lab_8
TYPE_ALab_8
TYPE_Lab_16
TYPE_Yxy_16
TYPE_YCbCr_8

TIEICC

TYPE_YCbCr_8_PLANAR
TYPE_YCbCr_16
TYPE_YCbCr_16_PLANAR
TYPE_YCbCr_16_SE
TYPE_YUV_8
TYPE_YUV_8_PLANAR
TYPE_YUV_16
TYPE_YUV_16_PLANAR
TYPE_YUV_16_SE
TYPE_HLS_8
TYPE_HLS_8_PLANAR
TYPE_HLS_16
TYPE_HLS_16_PLANAR
TYPE_HLS_16_SE
TYPE_HSV_8
TYPE_HSV_8_PLANAR
TYPE_HSV_16
TYPE_HSV_16_PLANAR
TYPE_HSV_16_SE
TYPE_XYZ_DBL
TYPE_Lab_DBL
TYPE_GRAY_DBL
TYPE_RGB_DBL
TYPE_CMYK_DBL

Unit HYIEUtils
IECC - Color Profile

Declaration

```
function Apply(SourceBitmap: TIEBitmap;
SourceFormat: integer; DestinationBitmap:
TIEBitmap; DestinationFormat: integer;
DestinationProfile: TIEICC; Intent: integer;
Flags: integer): boolean;
```

Description

Transforms a bitmap from current profile to the destination profile.
Look at Transform for parameters description.
You have to call FreeTransform after the whole image has been transformed. Apply returns false if it cannot perform the transformation.

TIEICC Supported Color Formats

Declaration

```
function Apply2(Bitmap: TIEBitmap; SourceFormat:
integer; DestinationFormat: integer;
DestinationProfile: TIEICC; Intent: integer;
Flags: integer): Boolean;
```

Description

Apply 2 transforms the same bitmap from current profile to the destination profile. Look at Transform for parameters description.
You have to call FreeTransform after the whole image has been transformed. Apply 2 returns false if it cannot perform the transformation.

TIEICC

Declaration

```
procedure Assign_LabProfile(WhitePoint_x,
WhitePoint_y, WhitePoint_Y_: double);
```

Description

Assign_LabProfile creates a new Lab profile based on specified white points.

Declaration

```
procedure Assign_LabProfileD50;
```

Description

Assign_LabProfileD50 creates a new Lab D59 color profile.

Declaration

```
procedure Assign_LabProfileFromTemp(TempK:
integer);
```

Description

Assign_LabProfileFromTemp creates a new Lab color profile based on specified temperature.

Unit HYIEUtils
IECC - Color Profile

DECLARATION

```
procedure Assign_sRGBProfile;
```

DESCRIPTION

Assign_sRGBProfile loads the predefined sRGB profile.

DECLARATION

```
procedure Assign_XYZProfile;
```

DESCRIPTION

Assign_XYZProfile creates a XYZ color profile.

DECLARATION

```
procedure Assign(source: TIEICC);
```

DESCRIPTION

Assign copies a profile from source.

TIEICC

IECC - Color Profile

DECLARATION

```
procedure Clear;
```

DESCRIPTION

Close the profile and free all allocated buffers.

DECLARATION

```
function ConvertBitmap(Bitmap: TIEBitmap;
DestPixelFormat: TIEPixelFormat; DestProfile:
TIEICC): Boolean;
```

DESCRIPTION

ConvertBitmap transforms the same bitmap from current profile to the destination profile. ConvertBitmap returns false if it cannot perform the transformation.

DECLARATION

```
property Copyright: AnsiString;
```

DESCRIPTION

Copyright returns a Copyright string found inside the ICC profile.

Unit HYIEUtils
IECC - Color Profile

DECLARATION

```
property Description: AnsiString;
```

DESCRIPTION

Description returns a ICC profile description found inside the ICC profile.

DECLARATION

```
procedure FreeTransform;
```

DESCRIPTION

Free memory allocated to perform Transform, Apply or Apply2.

DECLARATION

```
property IsApplied: Boolean;
```

DESCRIPTION

IsApplied returns true if Transform, Apply or Apply2 is executed since the profile has been loaded.

Unit HYIEUtils
IECC - Color Profile

DECLARATION

```
function IsValid: Boolean;
```

DESCRIPTION

IsValid returns true if the current loaded ICC profile is valid.

DECLARATION

```
procedure LoadFromBuffer(buffer: pointer;
bufferlen: integer);
```

DESCRIPTION

LoadFromBuffer loads the ICC profile from the specified buffer. This is called automatically when you load an image from file.

DECLARATION

```
procedure LoadFromFile(const FileName:string);
```

DESCRIPTION

LoadFromFile loads an ICC from the file. The file should have .icc extension.

TIEICC

Unit HYIEUtils
IECC - Color Profile

DECLARATION

```
procedure LoadFromStream(Stream: TStream;
StandardICC: Boolean);
```

DESCRIPTION

LoadFromStream loads an ICC from the stream. If StandardICC is False a header is expected to know the ICC block size.

DECLARATION

```
procedure SaveToStream(Stream: TStream;
StandardICC: Boolean);
```

DESCRIPTION

SaveToStream saves the current ICC to a stream. If StandardICC is False a header which specifies the ICC block size is added. Otherwise saves as standard ICC.

TIEICC

Unit HYIEUtils
IECC - Color Profile

DECLARATION

```
function Transform(Destination: TIEICC;
InputFormat: integer; OutputFormat: integer;
Intent: integer; Flags: integer; InputBuffer:
pointer; OutputBuffer: pointer; ImageWidth:
integer):Boolean;
```

DESCRIPTION

Transform transforms a row from current profile to the destination profile. You have to call FreeTransform after the whole image has been transformed. Transform returns false if it cannot perform the transformation. InputFormat and OutputFormat: The color format for the input and the output. Look below for a list of supported color formats. Intent: The ICC intent to apply. If an appropriate tag for this intent is not found, no error is raised and the intent is reverted to perceptual.

INTENT_PERCEPTUAL
INTENT_RELATIVE_COLORIMETRIC
INTENT_SATURATION
INTENT_ABSOLUTE_COLORIMETRIC

Flags: This value commands on how to handle the whole process. Some or none of this values can be joined via the 'or' operator.

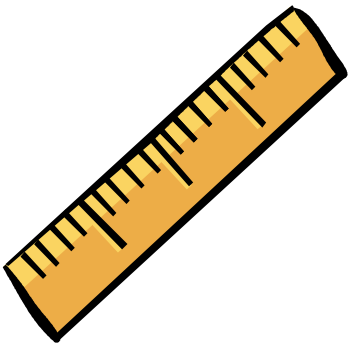
cmsFLAGS_MATRIXINPUT	CLUT ignored on input profile, matrix-shaper used instead (for speed, and debugging purposes)
cmsFLAGS_MATRIXOUTPUT	Same as anterior, but for output profile only.

cmsFLAGS_NOTPRECALC	By default, lcms smelt luts into a device-link CLUT. This speedup whole transform greatly. If you don't want this, and wish every value to be translated to PCS and back to output space, include this flag.
vcmsFLAGS_NULLTRANFORM	Don't transform anyway, only apply pack/unpack routines (useful to deactivate color correction but keep formatting capabilities)
cmsFLAGS_HIGHRESPRECALC	Use 48 points instead of 33 for device-link CLUT pre-calculation. Not needed but for the most extreme cases of mismatch of "impedance" between profiles.
cmsFLAGS_LOWRESPRECALC	Use lower resolution table. Useful when memory is a precious resource.
cmsFLAGS_BLACKPOINTCOMPENSATION	Use BPC algorithm.

Chapter 36

TRulerBox

Chapter 36. RulerBox



TRulerBox is a ruler. It was included in ImageEn for the histogram equalization preview dialog.

ImageEN
TRulerBox

Methods and Properties

Background	DotPerUnit	FitInView
Frequency	GripBaseDim	GripsColor
GripsCount	GripsDir	GripsMax
GripsMin	GripsPos	GripsKind
HexLabels	Inverted	LabelFreq
MaxGripHeight	RulerColor	RulerDir
Ruler	ScrollRate	ViewMax
ViewMin	ViewPos	

Events

OnRulerClick	OnRulerGripClick	OnRulerGripDblClick
OnRulerPosChange		

DECLARATION

```
property Background: TColor;
```

DESCRIPTION

Background is the background color.

RulerBox

DECLARATION

```
property DotPerUnit: double;
```

DESCRIPTION

DotPerUnit is the pixel for logical unit.

DECLARATION

```
property FitInView: Boolean;
```

DESCRIPTION

If FitInView is true, it specifies Automatic adjustment of DotPerUnit to fit ViewMin and ViewMax. FitInView adjusts DPU property using values of ViewMin and ViewMax.

DECLARATION

```
property Frequency: double;
```

DESCRIPTION

Frequency is the number of logical units where the ticks are shown.

RulerBox

DECLARATION

```
property GripBaseDim: integer;
```

DESCRIPTION

GripBaseDim is the base of the grip triangles in pixels.

DECLARATION

```
property GripsColor[g: integer]: integer;
```

DESCRIPTION

GripsColor sets color of the grip g.

DECLARATION

```
property GripsCount: integer;
```

DESCRIPTION

GripsCount gets/sets number of the grips.

Declaration

```
property GripsDir: TGripsDir;
```

Description

GripsDir is the direction of the grips (up or down).

Declaration

```
property GripsMax[g: integer]: double;
```

Description

GripsMax is the max value of the grip g.

Declaration

```
property GripsMin[g: integer]: double;
```

Description

GripsMin is the minimal value of the grip g.

DECLARATION

```
property GripsPos[g: integer]: double;
```

DESCRIPTION

GripsPos is the current position of the grip g.

DECLARATION

```
property GripsKind[g: integer]: TGripKind;
```

DESCRIPTION

GripsKind is the kind of the grip g (triangle, arrow, etc.).

DECLARATION

```
property HexLabels: Boolean;
```

DESCRIPTION

If HexLabels is true, the labels are shown in hexadecimal notation.

Declaration

```
property Inverted: Boolean;
```

Description

When true the ruler is painted from right to left.

Declaration

```
property LabelFreq: double;
```

Description

LabelFreq is the number of logical units where to show the labels.

Declaration

```
property MaxGripHeight: integer;
```

Description

MaxGripHeight specifies the maximum height of the grip.

DECLARATION

```
property OffsetX: integer;
```

DESCRIPTION

OffsetX specifies the horizontal offset where the ruler begins. The default is 0.

DECLARATION

```
property OffsetY: integer;
```

DESCRIPTION

OffsetY specifies the vertical offset where the ruler begins. Default is 0.

DECLARATION

```
property RulerColor: TColor;
```

DESCRIPTION

RulerColor is the color of the ruler.

RulerBox

Declaration

```
property RulerDir: TRulerDir;
```

Description

RulerDir is the direction of the ruler (horizontal or vertical).

Declaration

```
property Ruler: Boolean;
```

Description

If Ruler is true, the ruler is shown.

Declaration

```
property ScrollRate: double;
```

Description

ScrollRate specifies the scroll rate used when moving a grip out of ruler borders. The default is 1.

RulerBox

DECLARATION

```
property ViewMax: double;
```

DESCRIPTION

ViewMax is the maximum value for ViewPos property.

DECLARATION

```
property ViewMin: double;
```

DESCRIPTION

ViewMin is the minimum value for ViewPos property.

DECLARATION

```
property ViewPos: double;
```

DESCRIPTION

ViewPos is the initial position of viewing.

Events

DECLARATION

```
property OnRulerClick: TRulerClickEvent;
```

DESCRIPTION

OnRulerClick is called when the user clicks on the ruler.

DECLARATION

```
property OnRulerGripClick:
TRulerGripClickEvent;
```

DESCRIPTION

OnRulerGripClick is called when the user clicks on a grip.

RulerBox

ImageEN

TRulerBox

DECLARATION

```
property OnRulerGripDblClick:
TRulerGripDblClickEvent;
```

DESCRIPTION

This event occurs when the user double-click on a grip.

DECLARATION

```
property OnRulerPosChange:
TRulerPosChangeEvent;
```

DESCRIPTION

This event occurs when ViewPos changes.

RulerBox

Chapter 37

THSVBox

Chapter 37. HSVBox

THSVBox allows selection of a color in the HSV (Hue Saturation Value) color space.

ImageEN

THSVBox

Methods and Properties

Background	BarsDistance	Blue
Color	GetColorAt	Green
Hue	HueBarWidth	Red
Sat	SetColor	SetRGB
Val		

Events

OnChange

DECLARATION

```
property Background: TColor;
```

DESCRIPTION

Background is the background color.

HSVBox

DECLARATION

```
property BarsDistance: Integer;
```

DESCRIPTION

BarsDistance specifies the distance of Hue bar from color box (left box).

Example

HSVBox1.BarsDistance:=0; // removes distance from color bar and hue bar

DECLARATION

```
property Blue: byte;
```

DESCRIPTION

Blues is the blue channel of the conversion from current HSV color to RGB.

Read-only

ImageEN

THSVBox

DECLARATION

```
property Color: TColor;
```

DESCRIPTION

Color is the conversion of current HSV color in TColor.

Read-only

DECLARATION

```
function GetColorAt(x,y: integer): TColor;
```

DESCRIPTION

GetColorAt returns the color at component coordinates x, y. Useful in response to MouseMove event.

DECLARATION

```
property Green: byte;
```

DESCRIPTION

Green is the green channel of the conversion from current HSV color to RGB.

Read-only

HSVBox

ImageEN

THSVBox

DECLARATION

```
property Hue: integer;
```

DESCRIPTION

Hue is the hue channel of the current color.

DECLARATION

```
property HueBarWidth: integer;
```

DESCRIPTION

HueBarWidth specifies the width of the Hue bar. Set this value to 0 to remove the Hue bar (right bar).

Example

HSVBox1.HueBarWidth:=0; // removes Hue bar

HSVBox

ImageEN

THSVBox

DECLARATION

```
property Red: byte;
```

DESCRIPTION

Red is the red channel of the conversion from current HSV color to RGB.

Read-only

DECLARATION

```
property Sat: integer;
```

DESCRIPTION

Sat is the saturation channel of the current color.

DECLARATION

```
procedure SetColor(cl: TColor);
```

DESCRIPTION

SetColor sets current color as TColor.

HSVBox

ImageEN

THSVBox

DECLARATION

```
procedure SetRGB(r, g, b: byte);
```

DESCRIPTION

SetRGB sets current color as RGB.

DECLARATION

```
property Val: integer;
```

DESCRIPTION

Val is the value channel of the current color.

Events

DECLARATION

```
property OnChange: TNotifyEvent;
```

DESCRIPTION

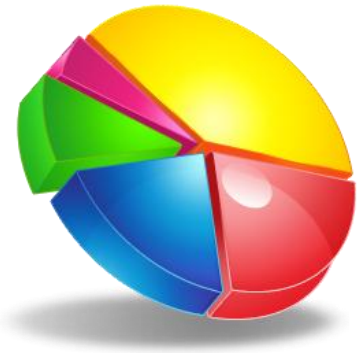
This event is called whenever a color is changed.

HSVBox

Chapter 38

TIEGradientBar

Chapter 38. IEGradientBar



Description

TIEGradientBar component shows a gradient bar, with a grip with which the user can select a color or an index.

ImageEN
TIEGradientBar

Methods and Properties

BeginColor	Color	ColorIndex
Direction	EndColor	MouseInteract
RGB		

Events

OnChange

DECLARATION

```
property BeginColor: TColor;
```

DESCRIPTION

BeginColor is the first color of the gradient bar.

DECLARATION

```
property Color: TColor;
```

DESCRIPTION

Color is the color the user has selected.

Read-only

IEGradientBar

DECLARATION

```
property ColorIndex: integer;
```

DESCRIPTION

ColorIndex is the color index that user has selected. It can be 0 (BeginColor) up to 255 (EndColor).

DECLARATION

```
property Direction: TIEGradientDir;
```

DESCRIPTION

Direction is gdHorizontal for a horizontal bar or gdVertical for a vertical bar.

DECLARATION

```
property EndColor: TColor;
```

DESCRIPTION

EndColor is the final color of the gradient bar.

ImageEN

TIEGradientBar

DECLARATION

```
property MouseInteract: TIEMouseInteractGr;
```

DESCRIPTION

Specify the automatic interactions with mouse (user).

MouseInteract can be migDragGrip if you want to select a color (or an index).

DECLARATION

```
property RGB: TRGB;
```

DESCRIPTION

The color the user has selected.

Read-only

IEGradientBar

Events

DECLARATION

```
property OnChange: TNotifyEvent;
```

DESCRIPTION

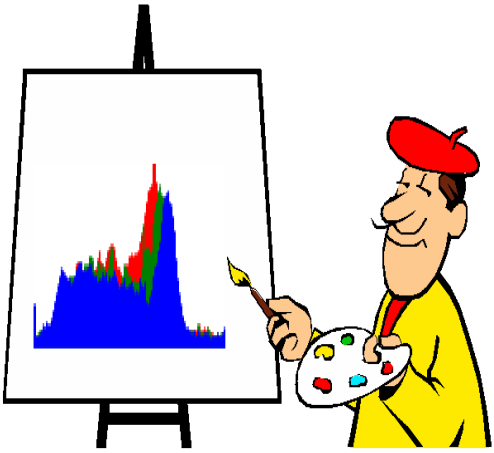
This event is called when the user selects a color.

IEGradientBar

Chapter 39

THistogramBox

Chapter 39. HistogramBox



DESCRIPTION

THistogramBox can be attached to a TImageEnProc component from which it gets information to compute and display the color channels histogram. THistogramBox is readonly.

ImageEN

THistogramBox

Methods and Properties

AttachedImageEnProc	Background	CompBar
GrayColor	HistogramKind	HistogramStyle
HistogramXPos	Labels	

Methods and Properties

DECLARATION

```
property AttachedImageEnProc: TImageEnProc;
```

DESCRIPTION

Use this property if you want to attach THistogramBox to a TImageEnProc object.

Example

```
HistogramBox1.AttachedImageEnProc :=
ImageEnProc1;
HistogramBox1.Update;
// now HistogramBox1
// display the histogram of image attached to
// ImageEnProc1
```

HistogramBox

Declaration

```
property Background: TColor;
```

Description

Background is the background color.

Declaration

```
property CompBar: Boolean;
```

Description

If True, THistogramBox shows a gradient bar at the bottom of the component.

Declaration

```
property GrayColor: TColor;
```

Description

GrayColor is the color assigned to the gray histogram.

DECLARATION

```
property HistogramKind: THistogramKind;
```

DESCRIPTION

Selects which channels are shown.

DECLARATION

```
property HistogramStyle:  THistogramStyle;
```

DESCRIPTION

HistogramStyle specifies if the histogram is displayed as lines or vertical bars.

DECLARATION

```
property HistogramXPos: integer;
```

DESCRIPTION

HistogramXPos is the left X coordinate where the histogram will be painted (with the component).

Read-only

HistogramBox

ImageEN
THistogramBox

DECLARATION

```
property Labels: THistogramLabels;
```

DESCRIPTION

Labels set which labels (vertical and horizontal) are shown.

HistogramBox

Chapter 40

Global Helper Functions

Chapter 40. Global Helper Functions

The author has developed hundreds of global helper functions over the years that assist with every day development needs. A few of these functions are included here to assist you developing with ImageEn.

Methods

Global Helper Functions

AddDefThousandSeparator	AddThousandSeparator
BitDepthToColorString	ColorToHex
ColorToHSL	ColorToPrettyName
ContrastingColor	DarkerColor
ExtractDate	ExtractRGB

ImageEN
Global Helper Functions

Global Helper Functions

ExtractTime

FormatStringByteSize

FormatStringKBSize

GetFileSize

GetRGB

HexToTColor

HSLToColor

IEFileTypeToGraphicFilterIndex

IntegerToString

JustFilename

JustName

LighterColor

SetGrayPal

SetSquarePen

TColorToHex

DECLARATION

```
function AddDefThousandSeparator ( const S:
string ): string;
// Adds the thousands separator from the
default locale in the correct location in a
string.
begin
  result := AddThousandSeparator ( S,
SysUtils.ThousandSeparator );
end;
```

DECLARATION

```
function AddThousandSeparator ( S: string;
Chr: Char ): string;
// Add a thousand separator to a string
var
  I: Integer;
begin
  result := S;
  I := length ( S ) - 2;
  while I > 1 do
  begin
    Insert ( Chr, result, I );
    I := I - 3;
  end;
end;
```

Declaration

```
function BitDepthToColorString ( ABitDepth:
integer ): string;
// Return a color string
begin
  if ABitDepth = 32 then
    Result := 'RGBA 32-Bit'
  else
    Result := 'RGB ' + IntToStr ( ABitDepth )
+ '-bit';
end;
```

Declaration

```
function ColorToHex (Color: TColor): string;
begin
  Result := IntToHex ( GetRValue (Color), 2) +
            IntToHex (GetGValue (Color), 2) +
            IntToHex(GetBValue(Color), 2);
end;
```

DECLARATION

```
procedure ColorToHSL (C: TColor; var H, S, L:
byte);
// Convert TColor to HSL
var
  Dif, CCmax, CCmin, RC, GC, BC, TempH, TempS,
TempL: Double;
begin
  C := ColorToRGB ( C );
  RC := GetRValue ( C ) / 255;
  GC := GetGValue ( C ) / 255;
  BC := GetBValue ( C ) / 255;
  if RC > GC then
    CCmax := RC
  else
    CCmax := GC;
  if BC > CCmax then
    CCmax := BC;
  if RC < GC then
    CCmin := RC
  else
    CCmin := GC;  if BC < CCmin then
    CCmin := BC;
  TempL := ( CCmax + CCmin ) / 2.0;
  if CCmax = CCmin then
  begin
    TempS := 0;
    TempH := 0;
  end
  else
  begin
    Dif := CCmax - CCmin;
```

```
    if TempL < 0.5 then
      TempS := Dif / ( CCmax + CCmin )
    else
      TempS := Dif / ( 2.0 - CCmax - CCmin );
    if RC = CCmax then
      TempH := ( GC - BC ) / Dif
    else if GC = CCmax then
      TempH := 2.0 + ( BC - RC ) / Dif
    else
      TempH := 4.0 + ( RC - GC ) / Dif;
    TempH := TempH / 6;
    if TempH < 0 then
      TempH := TempH + 1;
  end;
  H := Round ( 240 * TempH );
  S := Round ( 240 * TempS );
  L := Round ( 240 * TempL );
end;
```

Declaration

```
function ColorToRGBTriple ( const Color:
Graphics.TColor ): Windows.TRGBTriple;
{Converts a Delphi TColor value into an RGB
triple value.}
begin
  ExtractRGB ( Color, Result.rgbtRed,
    Result.rgbtGreen, Result.rgbtBlue );
end;
```

Declaration

```
function ColorToPrettyName (const AColor:
TColor): string;
{Return a string of a TColor from AColor}
var
  cs1: string;
begin
  Result := '';
  cs1 := ColorToString (AColor);
  if AnsiContainsText (cs1, 'cl') then
    Delete (cs1, 1, 2);
  Result := cs1;
end;
```

Declaration

```
function ContrastingColor (AColor: TColor):
TColor;
// Return s contracting color of the passed
// color
begin
  if ((AColor and $FF) * 77 + ((AColor shr 8)
and $FF) * 150 + ((AColor shr 16) and $FF) *
29) > 127 * 256 then
    Result := clBlack
  else
    Result := clWhite;
end;
```

Declaration

```
function DarkerColor (AColor: TColor;
AAdjustment: byte): TColor;
// Convert AColor to a darker color of same
// hue by AAdjustment value
var
  iH, iSS, iL: byte;
begin
  ColorToHSL (AColor, iH, iSS, iL);
  Result := HSLtoColor (iH, iSS, Max (iL -
    AAdjustment, 0));
end;
```

Declaration

```
function ExtractDate (TimeIn: TDateTime):
string;
  begin
    // Convert DateTime to long date string
    result := FormatDateTime('dddd, mmmm d,
      yyyy', TimeIn);
end;
```

DECLARATION

```
procedure ExtractRGB ( const Color:
Graphics.TColor; out Red, Green, Blue: Byte );
{Extracts constituent RGB values from a color.
System colors are converted to their actual
values per current Window display settings.}
var
  RGB: Windows.TColorRef; // RGB equivalent of
given Color

begin
  RGB := Graphics.ColorToRGB ( Color );
// ensures system Colors are converted
  Red := Windows.GetRValue ( RGB );
  Green := Windows.GetGValue ( RGB );
  Blue := Windows.GetBValue ( RGB );
end;
```

Declaration

```
// Convert DateTime to am/pm time string
function ExtractTime (TimeIn: TDateTime):
string;
begin
  result := FormatDateTime ('h:m AM/PM',
    TimeIn);
end;
```

Declaration

```
function FormatStringByteSize(TheSize: Int64
): string;
{Return an int64 as a string formatted similar
to the statusbar of Explorer}
var
  Buff: string;
begin
  SetLength (Buff, 255 * SizeOf(Char));
  ShLwApi.StrFormatByteSizeW (TheSize,
    PWideChar(Buff), 255 * SizeOf (Char));
  result := Trim (Buff);
end;
```

DECLARATION

```
function FormatStringKBSize (TheSize: Int64):
string;
{Return a formatted string of KBSize}
var
  Buff: string;
begin
  SetLength(Buff, 255 * SizeOf(Char));
  ShLwApi.StrFormatKBSizeW(TheSize, PWideChar
    (Buff), 255 * SizeOf(Char));
  result := Trim (Buff);
end;
```

Declaration

```
function GetFileSize (const Filename: string
): string;
var
  SearchRec: TSearchRec;
begin
  try
    if FindFirst (ExpandFileName (Filename
      ), faAnyFile, SearchRec) = 0 then
      result := FormatStringByteSize (
        SearchRec.Size)
    else
      result := '';
  finally
    SysUtils.FindClose (SearchRec);
  end;
end;
```

Declaration

```
procedure GetRGB ( Col: TColor; var R, G, B:
Byte );
var
  Color: $0..$FFFFFFFF;
begin
  Color := ColorToRGB ( Col );
  R := ( $000000FF and Color );
  G := ( $0000FF00 and Color ) shr 8;
  B := ( $00FF0000 and Color ) shr 16;
end;
```

ImageEN
Global Helper Functions

DECLARATION

```
function HexToTColor (AColor: string): TColor;
// Given Hex returns TColor from AColor
begin
  Result :=
    RGB(StrToInt('$' + Copy (AColor, 1, 2)),
       StrToInt ('$' + Copy (AColor, 3, 2)),
       StrToInt ('$' + Copy(AColor, 5, 2)));
end;
```

Declaration

```
function HSLToColor ( H, S, L: byte ): TColor;
// Convert HSL to TColor
var
  HN, SN, LN, RD, GD, BD, V, M, SV, Fract,
VSF, Mid1, Mid2: Double;
  R, G, B: byte;
  Sextant: integer;
begin
  HN := H / 239;
  SN := S / 240;
  LN := L / 240;
  if LN < 0.5 then
    V := LN * ( 1.0 + SN )
  else
    V := LN + SN - LN * SN;
  if V <= 0 then
  begin
    RD := 0.0;
    GD := 0.0;
    BD := 0.0;
  end
  else
  begin
    M := LN + LN - V;
    SV := ( V - M ) / V;
    HN := HN * 6.0;
    Sextant := Trunc ( HN );
    Fract := HN - Sextant;
    VSF := V * SV * Fract;
    Mid1 := M + VSF;
    Mid2 := V - VSF;
```

```
    case Sextant of
      0:
        begin
          RD := V;
          GD := Mid1;
          BD := M;
        end;
      1:
        begin
          RD := Mid2;
          GD := V;
          BD := M;
        end;
      2:
        begin
          RD := M;
          GD := V;
          BD := Mid1;
        end;
      3:
        begin
          RD := M;
          GD := Mid2;
          BD := V;
        end;
      4:
        begin
          RD := Mid1;
          GD := M;
          BD := V;
        end;
```

```
      5:
        begin
          RD := V;
          GD := M;
          BD := Mid2;
        end;
    else
      begin
        RD := V;
        GD := Mid1;
        BD := M;
      end;
    end;
  end;
  if RD > 1.0 then
    RD := 1.0;
  if GD > 1.0 then
    GD := 1.0;
  if BD > 1.0 then
    BD := 1.0;
  R := Round ( RD * 255 );
  G := Round ( GD * 255 );
  B := Round ( BD * 255 );
  Result := RGB ( R, G, B );
end;
```

DECLARATION

```
function IEFileTypeToGraphicFilterIndex (
const AIEFileType: TIOFileType ): integer;
// Returns the Graphic Filter Index from the
// AIEFileType as an integer
begin
  case AIEFileType of
    ioTIFF: Result := 1;
    ioGIF: Result := 2;
    ioJPEG: Result := 3;
    ioPCX: Result := 4;
    ioBMP: Result := 5;
    ioICO: Result := 6;
    ioPNG: Result := 7;
    ioWMF: Result := 8;
    ioEMF: Result := 9;
    ioTGA: Result := 10;
    ioPXM: Result := 11;
    ioJP2: Result := 12;
    ioJ2K: Result := 13;
    ioWBMP: Result := 14;
    ioDCX: Result := 15;
    ioHDP: Result := 16;
  else
    Result := -1;
  end;
end;
```

Declaration

```
function IntegerToString(const i: Int64):
string;
// Adds the locale default thousand separator
//in the correct location in a number and
// returns a string
begin
  Result := AddDefThousandSeparator IntToStr
    (i), SysUtils.ThousandSeparator );
end;
```

DECLARATION

```
function JustFilename ( const APathName:
string ): string;
// Return a filename from a string
var
  iString: string;
  i: Integer;
  // Reverse characters in a string ABCD ->
DCBA
  function Turn ( const AString: string ):
string;
  var
    i: Integer;
  begin
    Result := '';
    if AString <> '' then
      for i := 1 to Length (AString) do
        Result := AString [i] + Result;
  end;
begin
  iString := Turn (APathName);
  i := Pos ('\', iString);
  if i = 0 then
    i := Pos (':', iString );
  if i = 0 then
    Result := APathName
  else
    Result := Turn (Copy (iString, 1, i - 1));
end;
```

Declaration

```
function JustName (const APathName: string):
string;
// Return just the name from a file string
var
  iSS: string;
begin
  iSS := JustFilename (APathName);
  if Pos ('.', iSS) <> 0 then
    Result := Copy (iSS, 1, Pos ('.', iSS) -
1)
 else
    Result := iSS;
end;
```

DECLARATION

```
function LighterColor (C: TColor; AAdjustment:
byte): TColor;
// Convert AColor to a lighter color of same
// hue by AAdjustment value
var
  iH, iSS, iL: byte;
begin
  ColorToHSL ( C, iH, iSS, iL );
  Result := HSLtoColor ( iH, iSS, Min ( iL +
            AAdjustment, 255 ) );
end;
```

DECLARATION

```
procedure SetGrayPal ( hdc: integer );
var
  ip: array [ 0..255 ] of TRGBQuad;
  i: integer;
begin
  for i := 0 to 255 do
    with ip [ i ] do
    begin
      rgbRed := i;
      rgbGreen := i;
      rgbBlue := i;
      rgbReserved := 0;
    end;
  SetDibColorTable ( hdc, 0, 256, ip );
end;
```

Declaration

```
procedure SetSquarePen ( ACanvas: TCanvas;
AColor: TColor; AWidth: integer );
var
  iLogBrush: TLOGBRUSH;
begin
  if AWidth > 1 then
  begin
    iLogBrush.lbStyle := BS_Solid;
    iLogBrush.lbColor := AColor;
    iLogBrush.lbHatch := 0;
    ACanvas.Pen.Handle := ExtCreatePen (
      PS_Geometric or PS_Solid or
      PS_ENDCAP_SQUARE, AWidth, iLogBrush, 0,
      nil );

  end
  else
  begin
    ACanvas.Pen.Color := AColor;
    ACanvas.Pen.Width := AWidth;
    ACanvas.Pen.Style := psSolid;
  end;
end;
```

DECLARATION

```
function TColorToHex (AColor: TColor): string;
// given a TColor returns hex from AColor
begin
  Result :=
    IntToHex (GetRValue (AColor), 2) +
    IntToHex (GetGValue (AColor), 2) +
    IntToHex (GetBValue (AColor), 2);
end;
```

Chapter 41

Tips

Chapter 41. Tips

QUESTION:

How can I prevent the scrollwheel from causing the image to zoom? I want it to remain the original size at all times.

ANSWER:

```
ImageEnView1.MouseWheelParams.Action := iemwNone;
```

other values are iemwVScroll and iemwZoom (default).

ImageEN

Tips

QUESTION:

How do I convert a GIFImage to TIFF format without a visual component?

ANSWER:

There are several ways. The simplest is:

```
var
  iImageEnView: TImageEnView;
begin
  iImageEnView := TImageEnView.Create(nil);
  try
    iImageEnView.IO.LoadFromFile('in.gif');
    iImageEnView.IO.SaveToFile('out.gif');
  finally
    iImageEnView.Free;
  end;
end;
```

or use TImageEnIO:

```
var
iImageEnIO: TImageEnIO;
begin
  iImageEnIO := TImageEnIO.Create(nil);
  try
    iImageEnIO.LoadFromFile('in.gif');
    iImageEnIO.SaveToFile('out.gif');
  finally
    iImageEnIO.Free;
  end;
end;
```

You can also create a TIEBitmap (or TBitmap) object that contains the image as follows:

```
var
  iBitmap: TIEBitmap;
  iImageEnIO: TImageEnIO;
  begin
    iBitmap := TIEBitmap.Create;
    try
      iImageEnIO :=
      TImageEnIO.CreateFromBitmap(iBitmap);
    try
      iImageEnIO.LoadFromFile('in.gif');
      iImageEnIO.SaveToFile('out.gif');
    finally
      iImageEnIO.Free;
    end;
  finally
    iBitmap.Free;
  end;
end;
```

ImageEN
Tips

QUESTION:

Can I modify brightness?

ANSWER:

You can change brightness (luminosity) using several methods.
Using IntensityRGBAll method:

```
ImageEnView1.Proc.IntensityRGBall(20,20,20);
// increment luminosity of 20 (the fastest)
```

Using HSLvar method:

```
ImageEnView1.Proc.HSLvar(0,0,20); // increment
luminosity of 20 (slow but more accurate)
```

Using HSVvar method:

```
ImageEnView1.Proc.HSVvar(0,0,20);
// increment luminosity of 20 (slow but
accurate)
```

ImageEN
Tips

QUESTION:

Can I modify scanner resolution?

ANSWER:

To change scan resolution to 300 on your scanner use:

```
ImageEnView1.IO.TWainParams.XResolution.Curren
tValue := 300;
ImageEnView1.IO.TWainParams.YResolution.Curren
tValue := 300;
ImageEnView1.IO.Acquire;
```

QUESTION:

I'm trying to insert some text in the raw frame using the event OnVideoFrameRaw. How can I get a canvas from the parameters of this event?

ANSWER:

Instead of use OnVideoFrameRaw you have to use OnVideoFrame that returns a Bitmap object.
From Bitmap object you can obtain the Canvas with:

```
Bitmap.Canvas
```

ImageEN
Tips

QUESTION:

Can your component take a series of single page TIFF files and create a Multi-Page file from them?

ANSWER:

You can load the page using:

```
ImageEnView1.IO.LoadFromFile('page1.tif');
```

then save the page with:

```
ImageEnView1.IO.Params.TIFF_ImageIndex:=0; //
increment this for each page

ImageEnView1.IO.InsertToFileTIFF('multipage.tif');
```

Otherwise you can use TImageEnMView and TImageEnMIO component. See “multi” example for more details.

ImageEN
Tips

QUESTION:

How do I change the current page of a TIFF image in TImageEnView?

ANSWER:

In order to load several pages from a TIFF you have two ways:
1) load a page at the time, using TImageEnView, example:

```
ImageEnView1.IO.Params.TIFF_ImageIndex :=
page_number; // page_number starts from 0
(first page)

ImageEnView1.IO.LoadFromFile('mytiff.tiff');
```

First instruction select the page to load. To know how many pages there are use:

```
page_count := EnumTIFFIm('mytiff.tiff');
```

2) load all pages, using TImageEnMView. Just write:

```
ImageEnMView1.MIO.LoadFromFile('mytiff.tiff');
```

and you will see all pages.

ImageEN
Tips

QUESTION:

Is there any way to save a CMYK TIFF using ImageEn?

ANSWER:

To save TIFF with CMYK write:

```
ImageEnView1.IO.Params.TIFF_PhotometInterpret:
=ioTIFF_CMYK;
ImageEnView1.IO.SaveToFile('xxx.tif');
```

QUESTION:

When I try to resample a picture I get the compiler error : TResampleFilter is not defined!

ANSWER:

You have to add to the “uses” list the “hyiedefs” unit:

```
uses hyiedefs;
```

QUESTION:

Is it possible to de/activate duplex on scanner?

ANSWER:

Write ImageEnView1.IO.TWainParams.DuplexEnabled :=True (or False if you want disable).

ImageEN
Tips

QUESTION:

Is it possible to convert a multipage-image to a different encoding? G3 and G4 specifically?

ANSWER:

Using TImageEnMView you have to change the compression property for all pages:

```
ImageEnMView1.MIO.LoadFromFile('original.tif')
;
// change compression for the first page
ImageEnMView1.MIO.Params[0].TIFF_Compression
:= ioTIFF_G4FAX;
// change compression for the other pages
ImageEnMView1.MIO.DuplicateCompressionInfo;
// now save
ImageEnMView1.Mio.SaveToFile('output.tif');
```

ImageEN

Tips

QUESTION:

How do I get the transparent PNGs to work inside your vector view control?

ANSWER:

To load the alpha channel from PNG (and others) you have to use SetObjBitmapFromFile method:

```
ImageEnVect1.SetObjBitmapFromFile(hobj,
   'test.png');
```

QUESTION:

I would like users to be able to resize and move vectorial objects when they are displayed on the screen. Is it possible? How can I do that?

ANSWER:

To enable users to modify (move, resize) objects just set:

```
ImageEnVect1.MouseInteractVt :=
   [miObjectSelect];
```

The users can select then modify objects.

ImageEN
Tips

QUESTION:

How can I display images “on demand” with TImageEnMView?

ANSWER:

There are several ways to display images “on demand”:

1) If you have a directory where are all files just write:

ImageEnMView1.FillFromDirectory(‘c:\myimages’);

2) When you add a new image just set ImageFileName[] index, and ImageEn will load automatically specified file when needed.

```
Example:
Idx := ImageEnMView1.AppendImage;
ImageEnMView1.ImageFileName[idx]:=’first.jpg’;
```

3) When you add a new image just set the ImageID[] property. You have to create by hand an array of filenames where to get images.

Example:

```
var
files: array [0..1] of string;
begin
  files[0]:='first.jpg';
  files[1]:='second.jpg';
  ImageEnMView1.ImageID[ImageEnMView1.
    AppendImage] := 0;
  ImageEnMView1.ImageID[ImageEnMView1.
    AppendImage] := 1;
end;
```

You have also to create OnImageIDRequest event, on this you can write:

```
procedure
.OnImageIDRequest(Sender: TObject; ID:
integer; var Bitmap: TBitmap);
var
iImageEnIO: TImageEnIO;
begin
  iImageEnIO := TImageEnIO.Create(self);
  try
    iImageEnIO.AttachedBitmap := bmp;
    // bmp is a TBitmap object, defined at
    // class level (must exists after the
    // OnImageIDRequest exits)
    iImageEnIO.LoadFromFile(files[ID]);
    finally
      iImageEnIO.free;
    end;
  finally
    Bitmap := bmp;
  end;
end;
```

4) If the images are frames of a media file (like AVI, MPEG, etc.) you can write:

```
ImageEnMView1.LoadFromFileOnDemand('film.mpeg'
);
```

ImageEN
Tips

QUESTION:

How do I save (burn) lines, ellipses, etc. into the JPEG?

ANSWER:

Use CopyObjectsToBack method, then save the background image.

QUESTION:

How do I modify the IPTC fields without loading the original image?

ANSWER:

To load IPTC info from a jpeg, just use LoadFromFile method. After this you have in ImageEnView1.IO.Params.IPTC_Info object all IPTC information loaded. To read the caption you can write:

```
ImageEnView1.IO.LoadFromFile('image.jpg');
Idx :=
  ImageEnView1.IO.Params.IPTC_Info.
    IndexOf(2,120);
Caption := ImageEnView1.IO.Params.IPTC_Info.
  StringItem[id];
```

This modifies the caption:

```
ImageEnView1.IO.Params.IPTC_Info.StringItem[id
   x] := 'new caption';
```

```
ImageEnView1.IO.SaveToFile('image2.jpg');
```

If you want to modify IPTC info without load the image use ParamsFromFile and InjectJpegIPTC methods, in this way:

```
ImageEnView1.IO.ParamsFromFile('one.jpg');
```

... modify the IPTC info

```
ImageEnView1.IO.InjectJpegIPTC('two.jpg');
```

QUESTION:

How do I print all the images in a multipage TIFF?

ANSWER:

You can connect a TImageEnIO to a TImageEnMView. TImageEnIO.PrintImage will print the selected image (use SelectedImage to change current selected image). In this way you can print all pages:

```
for i := 0 to ImageEnMView1.ImageCount-1 do
begin
  ImageEnMView1.SeletedImage := i;
  ImageEnIO1.PrintImage...
end;
```

You can also use the predefined dialog of TImageEnMView component. Just write:

```
ImageEnMView1.MIO.DoPrintPreviewDialog;
```

ImageEN
Tips

QUESTION:

How I can save my jpeg without EXIF or other metadata?

ANSWER:

Call Params.ResetInfo. Example:

```
ImageEnView1.IO.LoadFromFile('input.jpg');
ImageEnView1.IO.Params.ResetInfo;
ImageEnView1.IO.SaveToFile('output.jpg');
```

QUESTION:

How can I check if the TImageEnView is empty (no bitmap loaded)?

ANSWER:

To empty the component use “Blank” method. To check if it is empty use IsEmpty:

```
if ImageEnView1.IsEmpty then
```

ImageEN
Tips

QUESTION:

When a user selects a page of the multipage TIFF image (using TImageEnMView) how do I display this image in a TImageEnView?

ANSWER:

To handle image selection use OnImageSelect event. To transfer current selected image use simply the Assign method:

```
procedure
TForm1.ImageEnMView1ImageSelect(Sender:
TObject; idx: Integer);
begin
  ImageEnView1.Assign(
    ImageEnMView1.Bitmap);
end;
```

QUESTION:

How do I assign the image in one ImageEnView to another ImageEnView?

ANSWER:

Just use Assign method:

```
ImageEnView1.Assign(ImageEnView2);
```

or

```
ImageEnView1.Assign(ImageEnView1.Bitmap);
// this doesn't copy DPI, but just the image
```

ImageEN
Tips

QUESTION:

Which is the correct way to load a Resource Image at runtime into a TImageEnView component, in C++?

ANSWER:

```
    TResourceStream *ResourceImage;
// Load from resource the About image ( a JPEG
file).
ResourceImage = new
TResourceStream((int)HInstance,
"ABOUTBITMAP",RT_RCDATA);
MainForm->ImageAbout->IO-
>LoadFromStreamJpeg(ResourceImage);
delete ResourceImage;
```

Here is a single line text file named "resource.rc" with the sentence:
ABOUTBITMAP RCDATA "about.jpg"
Just add the Resource file to the project and compile.

ImageEN
Tips

QUESTION:

How read custom TIFF tags?

ANSWER:

This example shows how read EXIF tags saved with Canon cameras:

```
var
ms: TMemoryStream;
tagReader1,tagReader2,tagReader3:
TIETifTagsReader;
i: integer;
// some Canon tags
m_nMacroMode,m_nLenghtTimer,m_Quality:
integer;
m_ImageType: string;
begin
  with ImageEnVect1 do
  begin
    IO.LoadFromFile('Capture_00006.JPG');
    with IO.Params.JPEG_MarkerList do begin
      i := IndexOf(JPEG_APP1);
      if i>=0 then begin
      // there are EXIF info
      ms := TMemoryStream.Create;
      ms.Write( MarkerData[i][6],
        MarkerLength[i] );
      // bypass first 4 bytes (must contain
         'Exif')
      ms.Position := 0;
      // read TIFF's IFD
      tagReader1:=
      TIETifTagsReader.CreateFromStream(ms,0);
      tagReader2 :=
      TIETifTagsReader.CreateFromIFD(
        tagReader1, 34665 );
       // read IFD in tag 34665 (SubEXIF)
      tagReader3 :=
      TIETifTagsReader.CreateFromIFD(
        tagReader2, $927c);
       // read IFD in tag $927C (MarkerData -
```

```
Canon IFD data)
      // read Canon EXIF tags
      m_nMacroMode :=
        tagReader3.GetIntegerIndexed(1,1);
      m_nLenghtTimer :=
        tagReader3.GetIntegerIndexed(1,2);
      m_Quality :=
        tagReader3.GetIntegerIndexed(1,3);
      m_ImageType := tagReader3.GetString(6);
      tagReader3.Free;
      tagReader2.Free;
      tagReader1.Free;
      ms.Free;
    end;
 end;
end;
```

ImageEN
Tips

QUESTION:

How do I compress my JPEGs to make them smaller?

ANSWER:

Jpeg is a file format with variable compression rate. The property that regulates the compression (and the quality) is JPEG_Quality. So you should set this property before save.

```
ImageEnView1.IO.LoadFromFile('input.jpg');
ImageEnView1.IO.Params.JPEG_Quality:=70;
ImageEnView1.IO.SaveToFile('output.jpg');
```

The default is 80, while other software uses 70.
If you want estimate the value used to create your file, execute this:

```
quality:=IECalcJpegFileQuality('input.jpg');
```

So you could write:

```
ImageEnView1.IO.LoadFromFile('input.jpg');
ImageEnView1.IO.Params.JPEG_Quality :=
  IECalcJpegFileQuality('input.jpg');
ImageEnView1.IO.SaveToFile('output.jpg');
```

If this is not enough, probably the file contains metatags (text info).
To remove them execute:

```
ImageEnView1.IO.Params.ResetInfo;
```

ImageEN
Tips

QUESTION:

How do I save an image with its objects when using TImageEnVect?

ANSWER:

Options of ImageEn:

1) If you want to save only objects:

```
ImageEnVect1.SaveToFileIEV('file.iev');
```

To load back:

```
ImageEnVect1.LoadfromFileIEV('file.iev');
```

2) If you want save objects and image:
and to load:

```
ImageEnVect1.LoadFromFileAll('file.all');

ImageEnVect1.SaveToFileAll('file.all');
```

3) If you want save objects and image as standard tiff and using Imaging Annotations (readable by Windows Preview):

```
ImageEnVect1.IO.SaveToFile('file.tif');
ImageEnVect1.SaveObjectsToTIFF('file.tif');
```

To load:

```
ImageEnVect1.IO.LoadFromFile('file.tif');
ImageEnVect1.LoadObjectsFromTIFF('
  file.tif');
```

4) If you want save objects inside a jpeg or other formats which do not support Imaging Annotations, or you just want to merge image and objects:

```
ImageEnVect1.CopyObjectsToBack;
ImageEnVect1.IO.SaveToFile('file.jpg');
```

ImageEN
Tips

QUESTION:

How do I store/retrieve multiple pages in a blob field using TImagenDBView?

ANSWER:

Tips

TImageEnDBView cannot store/retrieve multiple pages in a blob. However there is a solution. You can put a TImageEnMView component on the form and use it to load/save multiple pages as TIFFs in blobs using streams. To store the content of TImageEnMView inside a blob write:

```
var tempStream: TMemoryStream;
tempStream := TMemoryStream.Create;
ImageEnMView1.MIO.SaveToStreamTIFF(tempStream
);
BlobField.LoadFromStream(tempStream);
tempStream.free;
```

To retrieve from a blob:

```
tempStream := TMemoryStream.Create;
BlobField.SaveToStream(tempStream);
tempStream.Position := 0;
ImageEnMView1.MIO.LoadFromStreamTIFF(
  tempStream);
tempStream.free;
```

If you use a TDataSet inherited data set you can create a blob stream without using intermediate TMemoryStream: this will speed up operations.

```
BlobStream :=
myDataSet.CreateBlobStream(field, bmWrite);
ImageEnMView1.MIO.SaveToStreamTIFF(
  BlobStream );
  BlobStream.Free;
```

..and..

```
BlobStream :=
  myDataSet.CreateBlobStream(field,bmRead);
ImageEnMView1.MiO.LoadFromStreamTIFF(
BlobStream);
BlobStream.Free;
```

Finally, when save you can set compression info. For example, for black/white images you could write:

```
ImageEnMView1.MIO.Params[0].TIFF_Compression
:= ioTIFF_G4FAX;
ImageEnMView1.MIO.DuplicateCompressionInfo
```

ImageEN
Tips

QUESTION:

How do I load embedded JPEG images in a Raw file?

ANSWER:

Unfortunately Camera RAW formats aren't documented. Anyway it is possible to load embedded jpegs from NEF, CR2 and DNG raw formats.

Examples:

```
DNG:
  ImageEnView1.IO.Params.TIFF_SubIndex:=1;

ImageEnView1.IO.LoadFromFileTIFF('input.dng');
CR2:
  ImageEnView1.IO.Params.ImageIndex:=0;
ImageEnView1.IO.LoadFromFileTIFF('input.cr2');
NEF:
  ImageEnView1.IO.Params.ImageIndex:=1;

ImageEnView1.IO.LoadFromFileTIFF('input.nef');
CRW:
ImageEnView1.IO.LoadJpegFromFileCRW('input.crw
');
```

ImageEN

Tips

QUESTION:

How do I terminate a polyline without double-clicking?

ANSWER:

```
ImageEnVect.PolylineEndingMode :=
   ieemMouseUp;
```

QUESTION:

How do I read recent Camera RAW files?

ANSWER:

In order to read recent Camera RAW files you need to use the Dcraw plug-in.

QUESTION:

Measuring line length using miLineLen (or miArea) does not work. The mouse pointer only shows 0.00 pixels and nothing happens.

ANSWER:

miLineLen measures the perimeter of current selection. So a selection must be present. The same is for miArea which measures the area of current selection.
To measure the length of a line write instead:
`ImageEnVect1.MouseInteractVt:=[miDragLen];`

ImageEN
Tips

QUESTION:

I get “Floating point overflow” on printing.

ANSWER:

Sometime this happens on shared printers and depends by VCL or printer drivers bug. Try to disable FPU exceptions executing this before your printing code:

```
Set8087CW($133F);
```

QUESTION:

I see horizontal/vertical bands loading images using TImageEnMView. Why?

ANSWER:

By default, TImageEnMView loads EXIF thumbnails when they are available (with the thumbnail display method).
Sometimes these thumbnails contain black bands (horizontal or vertical bands).
To disable EXIF thumbnails loading set:

```
TImageEnMView.EnableLoadEXIFThumbnails :=
false;
```

ImageEN
Tips

QUESTION:

Sometimes I see bands after assigning bitmaps obtained from third-party components using TImageEnView. Why?

ANSWER:

The pixelformat of the assigned bitmap is incorrect. Try setting pixelformat after assigning the bitmap:

```
iBitmap := TBitmap.Create;
iBitmap.Assign (ThirdPartyComponent1.bitmap);
ImageEnView1.Bitmap.PixelFormat:= pf24bit;
ImageEnView.Update;
```

QUESTION:

When I open or assign 32-bit bitmaps to ImageEnView there is no transparency. Why?

ANSWER:

ImageEn has two properties that control how bitmaps are displayed: EnableAlphaChannel and IO.Params.BMP_HandleTransparency

Set EnableAlphaChannel to true and set IO.Params.BMP_HandleTransparency to true:

```
ImageEnView1.EnableAlphaChannel:= True;
ImageEnView1.IO.Params.BMP_HandleTransparenc
y:= True;
```

ImageEN
Tips

QUESTION:

How do I setup the Mouse Wheel?

ANSWER:

Set the MouseWheel.Params

```
// Mouse wheel will scroll image of 15 % of
component height

      ImageEnView1.MouseWheelParams.Action:=
      iemwVScroll;

      ImageEnView1.MouseWheelParams.Variation:=
      iemwPercentage;

      ImageEnView1.MouseWheelParams.Value:= 15;

// Set scrollbar params to match wheel

   ImageEnView1.HScrollBarParams.LineStep:= 15;
   ImageEnView1.VScrollBarParams.LineStep:= 15;
```

QUESTION:

How do I change selection grips?

ANSWER:

SetSelectionGripStyle

```
   ImageEnView1.SetSelectionGripStyle ( clBlack,
$00BAFFFF, bsSolid, 4, True, iegsCircle );
```

```
   ImageEnView1.SetLayersGripStyle ( clBlack,
$00BAFFFF, bsSolid, 4, iegsCircle );
```

QUESTION:

I can only do 5 undo's with ImageEnView. Why?

ANSWER:

The default limit for ImageEnView undo is 5. To increase the number of undo's set:

```
ImageEnView1.Proc.UndoLimit:= 99;
```

QUESTION:

When I display a grid in ImageEnView and add a layer, the grid is only visible on the Current Layer. Why?

ANSWER:

By default ImageEnView only displays a grid on the current layer. To display a grid on all layers set DisplayGridLyr to 0;

```
ImageEnView1.DisplayGridLyr:= 0;
```

ImageEN
Tips

QUESTION:

When I try to get the cursor position in ImageEnMouseMove the coordinates are incorrect. Why?

ANSWER:

The SelectionBase property sets the selection coordinates base. If SelectionBase is iesbClientArea (default), all coordinates depend upon actual zoom and scrolling. Otherwise, if SelectionBase is iesbBitmap, all coordinates refer to bitmap pixels. If the SelectionBase is set to iesbClientArea, then set the SelectionBase to iesbBitmap or convert the ClientArea selection coordinates to bitmap coordinates like this:

```
      X := ImageEnView.XScr2Bmp(X);
      Y := ImageEnView.YScr2Bmp(Y);
```

```
   If using layers use the layer conversion
methods:
X: = ImageEnView1.Layers
(ImageEnView1.CurrentLayer].ConvXScr2Bmp(X,
   Y);
Y: = ImageEnView1.Layers
(ImageEnView1.CurrentLayer].ConvYScr2Bmp(X,
   Y);
```