GetImageCount() or GetImageCountEx(). The usage of the function is described in detail at InsertImageEx().

Return values:

If the function succeeds the return value is the image handle, a value greater or equal zero. If the function fails the return value is a negative error code.

InsertMetafile

Syntax:

```
LBOOL pdfInsertMetafile(
   const void* IPDF,      // Instance pointer
   const char* FileName, // File path to EMF or WMF file
   double PosX,           // X-Coordinate of output rectangle
   double PosY,           // Y-Coordinate of output rectangle
   double Width,          // Width of output rectangle
   double Height)         // Height of output rectangle
```

The function converts an Enhanced Meta File (EMF) or Windows Meta File (WMF) to native PDF vector graphics. The function requires an open page or template into which the metafile can be drawn (see Append() or EditPage()). If a metafile should be used as fixed background on multiple pages, then insert it into a template and place this template on the pages where it should be used (see BeginTemplate() for further information). It is of course also possible to insert such a metafile on each page separately, but this would waste processing time and disk space.

The point *PosX*, *PosY* defines the lower left corner of the output rectangle if the coordinate system is bottom-up and the upper left corner otherwise.

Bounding box check

The function checks whether the resolution of the EMF file seems to be larger as 1800 DPI. If this is the case then the rclBounds rectangle is used to calculate the picture size since this is mostly an indication that the rclFrame rectangle was incorrectly calculated. If you need to process EMF files in a higher resoultion as 1800 DPI then disable the bounding box check with the flag mfNoBBoxCheck. See SetMetaConfFlags() for further information.

How to calculate the image size?

The calculation of the image size should be easy as possible. In most cases, metafiles must be inserted with exact proportions. Therefore, all metafile functions support two special values to make the calculation easier. The width and height can be calculated as follows:

- If *Width* or *Height* is -1 the function uses the original width or height from the image. If both parameters are -1 the metafile will be inserted with a resolution 72 DPI. However, this is not meaningful for metafiles because the logical size of a metafile is usually too large as if it could be used. You should not produce EMF files in a resolution of 72 DPI.

- If *Width* or *Height* is 0, the missing value is calculated in relation to the given value of *Height* or *Width* to preserve the metafile's aspect ratio. The resulting output is a vector graphic with exact proportions relative to its original size.
- If *Width* and *Height* is 0, the original size is used (same effect as -1, should not be used).
- Negative values of *Width* or *Height* are ignored; the function uses always the absolute values.

Font selection in EMF files

When selecting a font with the GDI function CreateFont(), the GDI selects a font that is a good match in comparison to the used parameters. The highest search priority has the Charset, followed by PitchAndFamily. If one of these parameters are not supported by the requested font, then the GDI selects an arbitrary other font that represents a better match.

To get a close match to what the GDI would select, the flag mfGDIFontSelection can be set (see SetMetaConvFlags() for further information). DynaPDF uses then a device context to select fonts. This produces often better results since many EMF files use invalid font names or other invalid parameters.

In general, DynaPDF checks whether the requested glyphs are available in the font. If one or more glyphs are missing then the selected font will be loaded in Unicode mode if it was loaded with a code page. If still glyphs are missing then the font will be changed to Arial Unicode MS. This fallback was mainly added for Asian EMF files which use often wrong charsets or fonts which do not support Asian characters.

Character sets

The charset parameter of a CreateFont() function should be set correctly so that DynaPDF can directly load a font with the required code page or Unicode:

- ANSI_CHARSET: The font will be loaded with the code page 1252. This saves disk space because ANSI strings require only one byte per character. However, if the same font must also be used with other encodings then use DEFAULT_CHARSET instead.
- DEFAULT_CHARSET: The font is directly loaded in Unicode mode. This charset must be used for any non-Latin charset.
- SYMBOL_CHARSET: This charset must be used for symbol fonts. Symbol codes start at 0xF000. The mapping in EMF file is often wrong. DynaPDF corrects this automatically.
- All other character sets are treated in Unicode mode. EMF files do not contain strings in other encodings.
- For Asian EMF files it is recommended but not required to set the flag mfUseUnicode. This flag makes sure that a font will not be loaded with the code page 1252 which is mostly not meaningful in Asian EMF files. It slightly improves the processing speed if a file contains many create font records with invalid charset information.

Non-portable WMF files

Two WMF file formats are available, the old non-portable WMF format and the newer portable WMF format. Both formats are converted to EMF before they can be converted to PDF. DynaPDF uses the Windows API function SetWinMetaFileBits() to convert WMF files. Because of this, WMF files are not supported on Mac OS X, Linux or UNIX.

However, the old non-portable WMF format is device depended; it contains no size information in the file's header so that this file type must be handled separately. The conversion function SetWinMetaFileBits() requires a parameter of the type METAFILEPICT, this structure is only initialized to default values by DynaPDF for non-portable WMF files:

```
METAFILEPICT pict;
pict.hMF  = NULL;
pict.mm   = MM_ANISOTROPIC;
pict.xExt = 0;
pict.yExt = 0;
```

The picture size is set to zero so that the GDI must calculate the size. WMF files converted in this way are often stretched. To get correct output results you must set the width and height of the WMF file manually with the function SetWMFDefExtent(). The size must be calculated in 0.01 millimetre units. A widely used size is 210000 x 280000 units for high resolution metafiles.

Notice: Missing lines or other objects indicates that the output size is too small. The GDI function SetWinMetaFileBits() removes records which would be invisible or too small to appear in the requested size. Set the output size to a larger value in this case.

Portable WMF files

The usage of portable WMF files is the same as of normal EMF files. However, many people want or must to personalize WMF files with additional contents and many of them have problems to calculate the correct WMF picture size which is required to get a correct EMF file. Let's see how DynaPDF calculates the size of a WMF file:

First, we need the header of the WMF file because the structure contains the size of the WMF file. The WMF header is defined as follows:

```
struct TRectS
{
   SI16 Left;
   SI16 Top;
   SI16 Right;
   SI16 Bottom;
};
#include <pshpack2.h>   // packed structure
struct TPORT_METAHEADER // 16 bit portable WMF file
{
   UI32   Key;      // WMF identifier (must be 0x9AC6CDD7)
   SI16   Handle;   // Number of handles in file
   TRectS BBox;     // Bounding rectangle
```

```
   UI16   Inch;     // Pixels per inch
   UI32   Reserved;
   UI16   CheckSum; // Aldus checksum
};
#include <poppack.h>
```

The header files pshpack2.h and poppack.h are available in Visual Studio and Embarcadeo's C++ Builder; they are used to declare a packed structure. Fill the structure TPORT_METAHEADER now with values:

```
TPORT_METAHEADER wmf;
fread(&wmf, 1, sizeof(wmf), f);
if (wmf.Key != 0x9AC6CDD7) // Is this a portable WMF file?
{
   fclose(f);
   return -2;
}
```

Now we can calculate the size of the metafile picture:

```
METAFILEPICT pict;
if (!wmf.Inch) wmf.Inch = 96;
pict.hMF  = NULL;
pict.mm   = MM_ANISOTROPIC;
pict.xExt = (wmf.BBox.Right - wmf.BBox.Left) * 2540 / wmf.Inch;
pict.yExt = (wmf.BBox.Bottom - wmf.BBox.Top) * 2400 / wmf.Inch;
```

As you can see above, the y-axis is not calculated with 2540 pixels per inch. DynaPDF uses 2400 pixels instead. I don't know why it must be 2400 pixels in most cases, but the y-axis is often stretched otherwise. This value can be adjusted in the range 2000 to 3000 with the property *WMFPixelPerInch*. This property changes the value of the y-axis only; the value of the x-axis is always 2540.

Notice: The adjustment above is maybe not required. The value 2400 pixels per inch was determined via trial and error.

ROP Codes (Raster Operation Codes)

EMF files are mostly created for a raster device like a monitor or printer. Because of this, metafiles use often ROP codes which combine fore- and background colors to achieve transparency or other color effects. Such ROP codes are meaningful on a raster devices but not on a vector device like PDF. Because no raster image is created during conversion, it is not possible to use ROP codes which combine fore- and background colors.

The function checks for unsupported ROP codes during conversion. If an unsupported ROP will be used then the EMF file will be rendered to an image by default. The problem with ROP codes is that the result depends on the background. So, the result will be often correct also if the file uses unsupported ROP codes. In most cases it is better to disable the rasterizer with the flag mfDisableRasterEMF and to enable it only if the output is wrong. See SetMetaConvFlags() for further information.

Notice:

When an EMF file is rendered to an image then it will be rendered in the resolution that was set with SetResolution(). The default resolution is 150 DPI which is usually too low to achieve good results. For good results the resolution should be set to 300 or 600 DPI.

How to convert spool EMF files?

EMF spool files contain EMF files for every page of a GDI print job. The spool file can also contain embedded fonts which are required by the EMF files. These fonts must be loaded in a pre-conversion step before the first EMF file can be converted. Embedded font subsets and corresponding delta font records must also be converted to regular TrueType fonts.

The function ConvertEMFSpool() can be used to convert a complete spool file to PDF, or if you want to use your own spool file parser, it can be used to load the embedded fonts from the spool file. Note that you must set the flags mfUseUnicode and mfIgnoreEmbFonts with SetMetaConvFlags() if only the spool fonts should be loaded. These flags are required in this case!

If you want to do anything manually with your own code then make sure that embedded font subsets and corresponding delta fonts will be converted back to regular TrueType fonts with Microsoft's function MergeFontPackage() of the fontsub.dll.

The resulting fonts as well as all other embedded fonts must be loaded with LoadFont() with the code page cpGlyphIndexes or cpUnicode (cpGlyphIndexes is preferred). In addition, the flags mfUseUnicode and mfIgnoreEmbFonts must be set with SetMetaConvFlags().

The code page cpGlyphIndexes or cpUnicode as well as the flag mfUseUnicode must be set to avoid collusions with the internal font selection during EMF conversion. The flag mfIgnoreEmbFonts makes sure that DynaPDF does not load the same embedded fonts again which are available in GDIComment records.

Compatibility Note:

DynaPDF versions prior 2.5.0.516 were not able to load spool fonts automatically. It was required to load the fonts in the user's temp directory manually with AddFontSearchPath() and the font cache had to be cleared before the next spool file could be converted. This technique was rather inefficient and is no longer supported. Existing applications should be changed to avoid the unnecessary calls of AddFontSearchPath() and ClearHostFonts(). AddFontSearchPath() does no longer load fonts with the extension tmp.

Remarks:

DynaPDF supports several flags to convert metafiles files to PDF, see SetMetaConvFlags() for further information. Because the GDI function SetWinMeatFileBits() is used to convert WMF files to EMF, WMF files are not supported on Mac OS X, Linux or UNIX.

Return values:

If the function succeeds the return value is 1. If the function fails the return value is 0.