NEWTONE

イメージ処理コンポーネント

ImageKit10 VCL

DLL コマンドリファレンス

目次

1. DLL コマンドリファレンス 3
1－1. Ik10Com.dll/Ik10ComA.dll/Ik10Com64.dll/Ik10Com64A.dll 4
1－2. Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll 31
1－3. Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll 93
1－4. Ik10Print.dll/Ik10PrintA.dll/Ik10Print64.dll/Ik10Print64A.dll 217
1－5. Ik10RasToVect.dll/Ik10RasToVectA.dll/Ik10RasToVect64.dll/Ik10RasToVect64A.dll 298
1－6. Ik10Scan.dll/Ik10ScanA.dll/Ik10Scan64.dll/Ik10Scan64A.dll 300
1－7. Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll 340
2. 作成したアプリケーションの配布 346
3. 以前の ImageKit との互換性 349
索引 350

1. DLL コマンドリファレンス

ImageKit10 には、VCL とほぼ同様な機能（描画系のコントロールは除く）を、DLL 形式のコマンドとしても用意してあります。

●ImageKit10 の DLL コマンドを使用する前に
呼び出し側のプログラムで DLL コマンド（プロシージャ）のプロトタイプ宣言の記述が必要になります。

開発環境に対応するヘッダファイルを参照してください。

コンテナ	ヘッダファイル名
C++Builder	ImageKit10.h
Delphi	ImageKit10.pas

ヘッダファイルは Ansi と Uniocde 共用となっているため、Ansi 環境(C++Builder 2007/Delphi 2007 以前)で Unicode 版の DLL を使用する場合や Unicode 環境(C++Builder 2009/Delphi 2009 以降)で Ansi 版の DLL を使用する場合はヘッダファイルの文字列の型を変更する必要があります(C++Builder 2009 以降でプロジェクトオプションの"_TCHAR のマップ先"を切り替える場合は変更不要です)。そのため、Ansi 環境では Ansi 版の DLL を Unicode 環境では Unicode 版の DLL を使用することをお勧めいたします。

●旧バージョンからの移行について

1)ImageKit5 からの移行
DLL 関数の名称と構造体名が"IK5・・・・"から"IK・・・"へ変更されています（既存の関数および構造体については・・・の部分は同じです）。その他にも関数に引数が追加されたり、新たに戻り値が追加されたりといくつかの変更が施されているものがあります。変更があるものについては【ImageKit5 との違い】という見出しで始まる簡単な説明がありますので、ご覧ください。
（注意）
Ik5Effect.dll の IK5ClearClipBrd,IK5CopyImage,IK5CreateImage,IK5GetFromClipBrd,IK5IsClipBrd,IK5SetToClipBrd は ImageKit10 では Ik10Com.dll/Ik10ComA.dll/Ik10Com64.dll/Ik10Com64A.dll に移動いたしました。

2)ImageKit6 からの移行
同じ構造体でも型が変更されたり、メンバー変数が追加されているもの、および同じ関数でも引数が変更されているものがあります。変更があるものについては【ImageKit6 との違い】という見出しで始まる簡単な説明がありますので、ご覧ください。
（注意）
Ik6Com.dll の IKCreateVectImage,IKCreateVectImageEx,IKDrawVectObject,IKVectToDib は ImageKit10 では Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll に移動いたしました。また、IKVectToDib は IKVectorToRaster に名称が変更されました。

3)ImageKit7 からの移行
同じ構造体でもメンバー変数が追加されているものや、Unicode への対応に伴い文字列型が変更されているものがあります。変更があるものについては【ImageKit7 との違い】という見出しで始まる簡単な説明がありますので、ご覧ください。

4)ImageKit8 からの移行
同じ構造体でもメンバー変数が追加されているものがあります。変更があるものについては【ImageKit8 との違い】という見出しで始まる簡単な説明がありますので、ご覧ください。

5)ImageKit9 からの移行
同じ構造体でもメンバー変数が追加されているものがあります。変更があるものについては【ImageKit9 との違い】という見出しで始まる簡単な説明がありますので、ご覧ください。

文章中に IK5,IK6,IK7,IK8,IK9,IK10 という表現がありますが、それぞれ ImageKit5・ImageKit6・ImageKit7・ImageKit8・ImageKit9・ImageKit10 の省略形を意味します。

1－1. Ik10Com.dll/Ik10ComA.dll/Ik10Com64.dll/Ik10Com64A.dll

ImageKit10 を使用する上で基本となる機能を提供します。

●DLL 関数コマンド一覧（アルファベット順）

関数名	内容
IKBitBlt	デバイスコンテキストのイメージを別のデバイスコンテキストへコピー
IKBitmapFromDib	DIB から DDB に変換
IKClearClipBrd	クリップボードのクリア
IKCopyImage	ラスタイメージやベクトルイメージの複製を作成
IKCreateImage	ラスタイメージを新規に作成
IKDeleteBitmapObject	ビットマップオブジェクトを削除
IKDibFromBitmap	DDB から DIB に変換
IKFreeMemory	メモリハンドルの解放
IKGetDpi	メモリハンドルから解像度（整数型）を取得
IKGetDpiF	メモリハンドルから解像度（浮動小数点型）を取得
IKGetDpiFromHdc	デバイスコンテキストから解像度を取得
IKGetErrorStatus	エラー番号を取得
IKGetFromClipBrd	クリップボードからラスタイメージやベクトルイメージの取り込み
IKGetImageType	メモリハンドルからイメージの情報を取得
IKGetMemorySize	メモリハンドルから使用しているイメージのメモリサイズを取得
IKGetOneBitPalCount	1 ビットカラーイメージのパレット 0 とパレット 1 のピクセル数を取得
IKGetPalette	メモリハンドルからパレットデータを取得
IKGetSystemPalette	現在の PC のシステムパレットを取得
IKIsClipBrdData	クリップボードのデータチェック
IKResizeRefine1BitImage	1 ビットイメージを 8 ビットグレースケールに補間してリサイズ
IKSetDpi	メモリハンドルへ解像度（整数型）を設定
IKSetDpiF	メモリハンドルへ解像度（浮動小数点型）を設定
IKSetPalette	メモリハンドルへパレットデータを設定
IKSetToClipBrd	ラスタイメージやベクトルイメージをクリップボードにコピー
IKStretchBlt	デバイスコンテキストのイメージを別のデバイスコンテキストへコピー（拡大縮小）

●構造体（ユーザ定義型）の定義および解説

IKIMAGE_INFO: イメージ情報を表します（**IKGetImageType** で使用します）。

(1)C++Builder

```
typedef struct {
    short       BitCount;
    short       PalCount;
    short       Mask1632;
    long        Xdpi;
    long        Ydpi;
    long        Width;
    long        Height;
    long        WidthByte;
    BOOL        Gray;
    long        ImageSize;
    short       ImageType;
} IKIMAGE_INFO;
typedef IKIMAGE_INFO * PTR_IKIMAGE_INFO;
```

(2)Delphi

```
type
    IKIMAGE_INFO = Record
        BitCount:           Smallint;
        PalCount:           Smallint;
```

```
    Mask1632:       Smallint;
    Xdpi:           Longint;
    Ydpi:           Longint;
    Width:          Longint;
    Height:         Longint;
    WidthByte:      Longint;
    Gray:           LongBool;
    ImageSize:      Longint;
    ImageType:      Smallint;
  end;
```

BitCount: イメージの 1 ピクセルあたりのビット数を表します。 *1
1:2 値、4:16 色、8:256 色、16:16 ﾋﾞｯﾄｶﾗｰ、24:24 ﾋﾞｯﾄｶﾗｰ、32:32 ﾋﾞｯﾄｶﾗｰ

PalCount: 使用パレット数を表します。(**BitCount** が 8 以下の場合が対象となり、それ以外は 0) *1
Mask1632: 16,32 ビットカラーの時のカラーマスクの種類を表します。 *1
0:**BitCount** が 16,32 でない、1:カラーマスクあり(RGB555 フォーマット)
2:カラーマスクあり(RGB565 フォーマット)、3:カラーマスクなし(RGB555 フォーマット)
4:カラーマスクあり(RGB888 フォーマット)、5:カラーマスクなし(RGB888 フォーマット)

Xdpi: イメージの横方向(X 方向)の 1 インチあたりのピクセル数を表します。
Ydpi: イメージの縦方向(Y 方向)の 1 インチあたりのピクセル数を表します。
Width: イメージの横のピクセル数を表します。
Height: イメージの縦のピクセル数を表します。
WidthByte: イメージの 1 ラインのバイト数を表します。 *1
Gray: True(0 以外):グレースケール False(0):グレースケールでない、を表します。 *1
ImageSize: イメージサイズのバイト数を表します。
ImageType: イメージのタイプを表します。
0:不明、1:ラスタイメージ、2:WMF、3:EMF、4:DXF、5:SVG、6:SXF

*1 ベクトルイメージの場合は使用しません。

IKIMAGE_INFO は、ImageKit5 の IK5IMAGE_INFO の後継にあたりますが、ImageType が新たに追加されています。

【ユーザ関数の定義】

【関数書式】
(1)C++Builder BOOL _stdcall UserProc(short Percent);
(2)Delphi function UserProc(Percent: Smallint): LongBool; stdcall;

【引数】

名称	内容
Percent	現在処理している%数

【戻り値】
False(0)の場合は実行している処理を終了します。True(0 以外)は処理を継続します。

ユーザ関数は UserProc という名称で説明しておりますが、実際の名称は何を設定されても構いません。関数名が UserProc の場合は以下のような形式で引数に渡します。
(1)C++Builder UserProc
(2)Delphi LONG_PTR(Addr(UserProc)) or LONG_PTR(@UserProc)

IKBitBlt

【機能】
コピー元からコピー先のデバイスコンテキストへ、指定された矩形内の各ピクセルの色データをコピーします。

【関数書式】
(1)C++Builder
BOOL IKBitBlt(HDC hDCDst, long XDst, long YDst, long Width, long Height, HDC hDCSrc, long XSrc, long YSrc, DWORD dwRop);
(2)Delphi
function IKBitBlt(hDCDst: HDC; XDst, YDst, Width, Height: Longint; hDCSrc: HDC; XSrc, YSrc: Longint; dwRop: DWORD): LongBool;

【引数】

名称	内容
hDCDst	コピー先のデバイスコンテキスト
XDst	コピー先矩形の左上隅の x 座標(ピクセル)
YDst	コピー先矩形の左上隅の y 座標(ピクセル)
Width	コピー先矩形の幅(ピクセル)
Height	コピー先矩形の高さ(ピクセル)
hDCSrc	コピー元のデバイスコンテキスト
XSrc	コピー元矩形の左上隅の x 座標(ピクセル)
YSrc	コピー元矩形の左上隅の y 座標(ピクセル)
dwRop	ラスタオペレーションコード

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
ラスタオペレーションモードはデバイスコンテキストにどのようにコピーするかを指定します。

値	説明
0x00000042	コピー先の矩形を黒色で塗りつぶします。(BLACKNESS)
0x00550009	コピー先の矩形の色を反転します。(DSTINVERT)
0x00C000CA	論理 AND 演算子を使って、コピー元の色とコピー先の色を組み合わせます。(MERGECOPY)
0x00BB0226	論理 OR 演算子を使って、コピー元の色を反転した色と、コピー先の色を組み合わせます。(MERGEPAINT)
0x00330008	コピー元の色を反転して、コピー先へコピーします。(NOTSRCCOPY)
0x001100A6	論理 OR 演算子を使って、コピー元の色とコピー先の色を組み合わせ、さらに反転します。(NOTSRCERASE)
0x00F00021	指定したパターンをコピー先へコピーします。(PATCOPY)
0x005A0049	論理 XOR 演算子を使って、指定したパターンの色と、コピー先の色を組み合わせます。(PATINVERT)
0x00FB0A09	論理 OR 演算子を使って、指定したパターンの色と、コピー元の色を反転した色を組み合わせます。さらに論理 OR 演算子を使って、その結果と、コピー先の色を組み合わせます。(PATPAINT)
0x008800C6	論理 AND 演算子を使って、コピー元の色とコピー先の色を組み合わせます。(SRCAND)
0x00CC0020	コピー元の矩形をコピー先の矩形へそのままコピーします。(SRCCOPY)
0x00440328	論理 AND 演算子を使って、コピー先の色を反転した色と、コピー元の色を組み合わせます。(SRCERASE)
0x00660046	論理 XOR 演算子を使って、コピー元の色とコピー先の色を組み合わせます。(SRCINVERT)
0x00EE0086	論理 OR 演算子を使って、コピー元の色とコピー先の色を組み合わせます。(SRCPAINT)
0x00FF0062	コピー先の矩形を白色で塗りつぶします。(WHITENESS)

通常は 0x00CC0020(SRCCOPY)を指定します。
※16 進表記のため、Delphi は 0x を$に置き換えてください。
()内の説明は WindowsAPI で使用する定数と同じ意味です。

IKBitmapFromDib

【機能】
デバイス独立ビットマップ（DIB）からデバイス依存ビットマップ（DDB）に変換します。

【関数書式】
(1)C++Builder
HBITMAP IKBitmapFromDib(HANDLE Handle);
(2)Delphi
function IKBitmapFromDib(Handle: THandle): HBitmap;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル

【戻り値】
ビットマップハンドル（実行に失敗した場合は 0 もしくは NULL）

【解説】
C++Builder/Delphi の TBitmap へのイメージの受け渡しなどに使用します。引数として与えた Handle は解放されずにそのまま残ります。

Delphi の例：

```
{ TBitmap にラスタイメージを設定 }
var
  Bitmap1: TBitmap;
  ImageHandle: THandle;
begin
  ImageHandle := IKFileLoad('C:¥Png¥001.png', 0, 0, 0, 0, nil, nil, nil);
  Bitmap1 := TBitmap.Create;
  try
    Bitmap1.Handle := IKBitmapFromDib(ImageHandle);
    { イメージ処理 }
  finally
    Bitmap1.Free;
    IKFreeMemory(ImageHandle);
  end;
end;
```

IKClearClipBrd

【機能】
クリップボードをクリアします。

【関数書式】
(1)C++Builder
void IKClearClipBrd(HWND hWnd);
(2)Delphi
procedure IKClearClipBrd(hWnd: HWND);

【引数】

名称	内容
hWnd	呼び出す側のウィンドウハンドル

【戻り値】
ありません。

IKCopyImage

【機能】
メモリ上のラスタイメージやベクトルイメージを別のメモリ上にコピーします。

【関数書式】
(1)C++Builder
HANDLE IKCopyImage(HANDLE Handle);
(2)Delphi
function IKCopyImage(Handle: THandle): THandle;

【引数】

名称	内容
Handle	コピー元のラスタイメージやベクトルイメージのメモリハンドル (ラスタイメージは 1,4,8,16,24,32,48 ビットイメージが対象)

【戻り値】
コピー先のラスタイメージやベクトルイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
IKCopyImage を実行しても、コピー元のイメージはそのまま残ります。
48 ビットイメージは Ik10Scan.dll/Ik10ScanA.dll/Ik10Scan64.dll/Ik10Scan64A.dll のいずれかを使用してスキャンデバイスから 36,42,48 ビットカラーで取り込んだイメージのコピーなどに使用します。

(注意)
引数の Handle にベクトルイメージを指定する場合は、
Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll のいずれかが別途必要です。

【ImageKit6 との違い】
引数 UserProc,Caption,Message,Button が削除されました。

IKCreateImage

【機能】
新規にラスタイメージを作成します。

【関数書式】
(1)C++Builder
HANDLE IKCreateImage(long Width, long Height, short BitCount, BYTE Red, BYTE Green, BYTE Blue, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKCreateImage(Width, Height: Longint; BitCount: Smallint; Red, Green, Blue: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Width	作成するイメージの幅(ピクセル)
Height	作成するイメージの高さ(ピクセル)
BitCount	作成したいイメージのビット数(1,4,8,16,24,32,40,80) ※4 は 4 ビットカラー、40 は 4 ビットグレー、8 は 8 ビットカラー、80 は 8 ビットグレーとなります。
Red	作成するイメージカラーの赤 (0～255)
Green	作成するイメージカラーの緑 (0～255)
Blue	作成するイメージカラーの青 (0～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
作成したラスタイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
ラスタイメージをメモリ上に作成します。
作成するイメージのビット数により、設定した RGB 値が有効にならない場合は、その値に一番近い値を割り当てます。(24 ビット以上のイメージの場合は設定した値がそのまま有効となります。)

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Com.dll/Ik10ComA.dll/Ik10Com64.dll/Ik10Com64A.dll」のユーザ関数の定義**をご覧ください。

IKDeleteBitmapObject

【機能】
ビットマップオブジェクトを削除します。

【関数書式】
(1)C++Builder
BOOL IKDeleteBitmapObject(HBITMAP hBm);
(2)Delphi
function IKDeleteBitmapObject(hBm: HBitmap): LongBool;

【引数】

名称	内容
hBm	ビットマップハンドル

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
IKBitmapFromDib で取得したビットマップハンドルが不要になった場合に使用します。

IKDibFromBitmap

【機能】
デバイス依存ビットマップ（DDB）からデバイス独立ビットマップ（DIB）に変換します。

【関数書式】
(1)C++Builder
```
HANDLE IKDibFromBitmap(HBITMAP hBm, HPALETTE hPal);
```
(2)Delphi
```
function IKDibFromBitmap(hBm: HBitmap; hPal: HPalette): THandle;
```

【引数】

名称	内容
hBm	ビットマップハンドル
hPal	ビットマップのパレット（パレットがない場合は 0）

【戻り値】
ラスタイメージのメモリハンドル（実行に失敗した場合は 0 もしくは NULL）

【解説】
C++Builder/Delphi の TBitmap からイメージを取得する場合などに使用します。引数として与えた hBm は削除されません。

Delphi の例：
```
{ TBitmap からラスタイメージを取得 }
var
  Bitmap1: TBitmap;
  ImageHandle: THandle;
begin
  Bitmap1 := TBitmap.Create;
  try
    Bitmap1.LoadFromFile('C:¥Bmp¥001.bmp');
    { イメージ処理 1 }
    ImageHandle := IKDibFromBitmap(Bitmap1.Handle, Bitmap1.Palette);
    { イメージ処理 2 }
  finally
    Bitmap1.Free;
  end;
end;
```

IKFreeMemory

【機能】
イメージデータが占有するメモリを解放します。

【関数書式】
(1)C++Builder
BOOL IKFreeMemory(HANDLE Handle);
(2)Delphi
function IKFreeMemory(Handle: THandle): LongBool;

【引数】

名称	内容
Handle	解放するイメージデータのメモリハンドル（ラスタイメージとベクトルイメージおよび Raw データが対象）

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
使用可能なメモリサイズには限りがありますので、適宜不要になったイメージのメモリを解放してください。
また、メモリの 2 重解放を防ぐため、関数実行後引数の Handle に 0 を設定するようにしてください。
Delphi の例：

```
IKFreeMemory(Handle);
Handle := 0;
```

（注意）
引数の Handle にベクトルイメージを指定する場合は、**Ik10Com.dll/Ik10ComA.dll/Ik10Com64.dll/Ik10Com64A.dll** が別途必要です。

IKGetDpi,IKGetDpiF

【機能】
イメージデータの解像度を取得します。

【関数書式】
(1)C++Builder
BOOL **IKGetDpi**(HANDLE Handle, long *Xdpi, long *Ydpi);
BOOL **IKGetDpiF**(HANDLE Handle, float *Xdpi, float *Ydpi);
(2)Delphi
function **IKGetDpi**(Handle: THandle; var Xdpi, Ydpi: Longint): LongBool;
function **IKGetDpiF**(Handle: THandle; var Xdpi, Ydpi: Single): LongBool;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル
Xdpi	取得するX(横)方向の1インチあたりのピクセル数
Ydpi	取得するY(縦)方向の1インチあたりのピクセル数

【戻り値】
実行に失敗した場合はFalse(0)、成功した場合はTrue(0以外)が返されます。

【解説】
成功した場合、XdpiとYdpiに解像度が設定されます。

IKGetDpiFromHdc

【機能】
デバイスコンテキストの解像度を取得します。

【関数書式】
(1)C++Builder
BOOL IKGetDpiFromHdc(HDC hDC, long *Xdpi, long *Ydpi);
(2)Delphi
function IKGetDpiFromHdc(hDC: HDC; var Xdpi, Ydpi: Longint): LongBool;

【引数】

名称	内容
hDC	デバイスコンテキスト
Xdpi	取得するX(横)方向の1インチあたりのピクセル数
Ydpi	取得するY(縦)方向の1インチあたりのピクセル数

【戻り値】
実行に失敗した場合はFalse(0)、成功した場合はTrue(0以外)が返されます。

【解説】
スクリーンやプリンタなどのデバイスコンテキストの解像度を取得します。
成功した場合、XdpiとYdpiに解像度が設定されます。

IKGetErrorStatus

【機能】
エラー番号を取得します。

【関数書式】
(1)C++Builder　int IKGetErrorStatus(void);
(2)Delphi　function IKGetErrorStatus: Integer;

【引数】
ありません。

【戻り値】

値	内容
0	正常終了(エラーなし)
1	キャンセル
2	その他のエラー
3	メモリエラー
4	サポート外機能
5	設定値が不正
6	ファイルが不正
7	イメージが不正
8	Twain_32.dll または TWAINDSM.dll がロードできない
9	描画エラー
10	対象外イメージ
11	実行順序エラー
12	ADF に用紙がセットされていない(UI 非表示のみ)
13	デジタルカメラに画像がない(UI 非表示のみ)
14	DLL がロードされていない
15	通信エラー
16	ファイルが見つからない
17	ファイルが壊れている
18	タイムオーバーで処理をキャンセル
19	パスワード、ユーザ名が異なる(アクセスができない)
20	ファイルがオープンできない
21	ファイルが作成できない
22	ファイルが書き込みできない
23	ファイルが読み込みできない
24	表示イメージデータがない
25	PDF に設定したパスワードのエラー
101	Twain_32.dll または TWAINDSM.dll にエントリポイントがない
102	アプリケーションハンドルもしくはソースハンドルが不正
103	データソースがない
104	データソースは既に最大数のアプリケーションに接続されている
105	データソースまたはデータソースマネージャでエラーが発生
106	認識できないトリプレット(TWAIN - DG_XXX/DAT_XXX/MSG_XXX)
107	スキャンデバイスの稼動状態を確認
108	フィーダーに用紙が詰まった
109	フィーダーに複数の用紙が取り込まれた
110	解像度が不正
111	カバーが開いている
112	原稿の角が傷んでいる
113	原稿をキャプチャー中のフォーカスエラー
114	原稿がとても明るい
115	原稿がとても暗い
116	イメージ情報が取得できないため転送不可

【解説】

ImageKit10 の DLL コマンド内で起きた直前のエラー番号を返します。
戻り値が 13 の場合は、デジタルカメラの TWAIN ドライバが TWAIN 規約のバージョン 1.7 以降に対応している場合が対象で、108・109 は TWAIN ドライバが TWAIN 規約のバージョン 1.8 以降に対応している場合が対象となります。
戻り値が 110 の場合はエプソン製スキャナ用ドライバご利用時にモアレ除去機能を使用した場合が対象です。
戻り値が 111～115 は TWAIN ドライバが TWAIN 規約のバージョン 2.0 以降に対応している場合が対象で、116 は TWAIN ドライバが TWAIN 規約のバージョン 2.1 以降に対応している場合が対象となります。

IKGetFromClipBrd

【機能】
クリップボードからラスタイメージやベクトルイメージを取り込みます。

【関数書式】
(1)C++Builder
HANDLE IKGetFromClipBrd(HWND hWnd, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKGetFromClipBrd(hWnd: HWND; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
hWnd	呼び出す側のウィンドウハンドル
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ラスタイメージやベクトルイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
取得するイメージがベクトルの場合、EMF 形式として扱います。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、「Ik10Com.dll/Ik10ComA.dll/Ik10Com64.dll/Ik10Com64A.dll」のユーザ関数の定義をご覧ください。

IKGetImageType

【機能】
イメージ情報を取得します。

【関数書式】
(1)C++Builder
BOOL IKGetImageType(HANDLE Handle, PTR_IKIMAGE_INFO ImageInfo);
(2)Delphi
function IKGetImageType(Handle: THandle; var ImageInfo: IKIMAGE_INFO): LongBool;

【引数】

名称	内容
Handle	イメージデータのメモリハンドル(ラスタイメージとベクトルイメージが対象)
ImageInfo	取得するイメージ情報の構造体(ユーザ定義型)変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
成功した場合、ImageInfo にイメージ情報が設定されます。
IKIMAGE_INFO については、**「Ik10Com.dll/Ik10ComA.dll/Ik10Com64.dll/Ik10Com64A.dll」の IKIMAGE_INFO のメンバー変数の説明**をご覧ください。

※ImageInfo の構造は IK5 と IK6 以降では異なります(IK6 以降は同じ)。

IKGetMemorySize

【機能】
イメージデータが使用しているメモリサイズを取得します。

【関数書式】
(1)C++Builder
DWORD IKGetMemorySize(HANDLE Handle);
(2)Delphi
function IKGetMemorySize(Handle: THandle): DWORD;

【引数】

名称	内容
Handle	イメージデータのメモリハンドル(ラスタイメージとベクトルイメージおよび Raw データが対象)

【戻り値】
使用しているメモリサイズが返されます(バイト単位)。実行に失敗した場合は 0 が返されます。

IKGetOneBitPalCount

【機能】
1 ビットカラーイメージのパレット 0 とパレット 1 のピクセル数を取得します。

【関数書式】
(1)C++Builder
BOOL IKGetOneBitPalCount(HANDLE Handle, long *Pal0, long *Pal1);
(2)Delphi
function IKGetOneBitPalCount(Handle: THandle; var Pal0, Pal1: Longint): LongBool;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル
Pal0	取得する 1 ビットカラーイメージのパレット 0 のピクセル数
Pal1	取得する 1 ビットカラーイメージのパレット 1 のピクセル数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
Handle の示すイメージは 1 ビットカラーイメージが対象です。
成功した場合、Pal0,Pal1 にピクセル数が設定されます。
パレットの色については、**IKGetPalette** を参照してください。

IKGetPalette

【機能】
ラスタイメージのパレット情報を取得します。

【関数書式】
(1)C++Builder
BOOL IKGetPalette(HANDLE Handle, short *Red, short *Green, short *Blue);
(2)Delphi
function IKGetPalette(Handle: THandle; var Red, Green, Blue: Smallint): LongBool;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル
Red	取得する赤のパレット情報の配列
Green	取得する緑のパレット情報の配列
Blue	取得する青のパレット情報の配列 (1)C++Builder　配列の先頭のポインタを渡す (2)Delphi　引数に Red[0]のように与える

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
パレット情報を取得する RGB のそれぞれの配列の数は IKIMAGE_INFO の PalCount になります。
PalCount は **IKGetImageType** で取得できます。
IKIMAGE_INFO については、**「Ik10Com.dll/Ik10ComA.dll/Ik10Com64.dll/Ik10Com64A.dll」の IKIMAGE_INFO のメンバー変数の説明**をご覧ください。

IKGetSystemPalette

【機能】
現在の PC のシステムパレットを取得します。

【関数書式】
(1)C++Builder
int IKGetSystemPalette(void);
(2)Delphi
function IKGetSystemPalette: Integer;

【引数】
ありません。

【戻り値】

	内容
	現在のコンピュータのシステムパレットを返します。 2:2 色(1 ビットカラー) 16:16 色(4 ビットカラー) 256:256 色(8 ビットカラー) 0:16 ビットカラー、24 ビットカラー、32 ビットカラー

IKIsClipBrdData

【機能】
クリップボードにデータがあるかどうかをチェックします。

【関数書式】
(1)C++Builder
BOOL IKIsClipBrdData(HWND hWnd);
(2)Delphi
function IKIsClipBrdData(hWnd: HWND): LongBool;

【引数】

名称	内容
hWnd	呼び出す側のウィンドウハンドル

【戻り値】
データがない場合は False(0)、データがある場合は True(0 以外)が返されます。

IKResizeRefine1BitImage

【機能】
1ビットイメージを8ビットグレースケールに補間してリサイズします。

【関数書式】
(1)C++Builder
HANDLE IKResizeRefine1BitImage(HANDLE Handle, long Width, long Height);
(2)Delphi
function IKResizeRefine1BitImage(Handle: THandle; Width, Height: Longint): THandle;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル
Width	処理後のイメージの幅(ピクセル)
Height	処理後のイメージの高さ(ピクセル)

【戻り値】
ラスタイメージのメモリハンドル(実行に失敗した場合は、0が返されます)

【解説】
1ビットイメージを高精彩に縮小してイメージを作成します。
引数として与えたメモリハンドルは解放されずにそのまま残ります。

IKSetDpi,IKSetDpiF

【機能】
イメージデータの解像度を設定します。

【関数書式】
(1)C++Builder
BOOL **IKSetDpi**(HANDLE Handle, long Xdpi, long Ydpi);
BOOL **IKSetDpiF**(HANDLE Handle, float Xdpi, float Ydpi);
(2)Delphi
function **IKSetDpi**(Handle: THandle; Xdpi, Ydpi: Longint): LongBool;
function **IKSetDpiF**(Handle: THandle; Xdpi, Ydpi: Single): LongBool;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル
Xdpi	設定するX(横)方向の1インチあたりのピクセル数
Ydpi	設定するY(縦)方向の1インチあたりのピクセル数

【戻り値】
実行に失敗した場合はFalse(0)、成功した場合はTrue(0以外)が返されます。

【解説】
成功した場合、XdpiとYdpiをHandleの示すイメージの解像度に設定します。

IKSetPalette

【機能】
ラスタイメージのパレット情報を設定します。

【関数書式】
(1)C++Builder
BOOL IKSetPalette(HANDLE Handle, short *Red, short *Green, short *Blue);
(2)Delphi
function IKSetPalette(Handle: THandle; var Red, Green, Blue: Smallint): LongBool;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル
Red	設定する赤のパレット情報の配列
Green	設定する緑のパレット情報の配列
Blue	設定する青のパレット情報の配列 (1)C++Builder　配列の先頭のポインタを渡す (2)Delphi　引数に Red[0]のように与える

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
設定できるパレット数はイメージのパレット数と同じでなければなりません。パレット情報を設定する RGB のそれぞれの配列の数は IKIMAGE_INFO の PalCount になります。PalCount は **IKGetImageType** で取得できます。
IKIMAGE_INFO については、**「Ik10Com.dll/Ik10ComA.dll/Ik10Com64.dll/Ik10Com64A.dll」の IKIMAGE_INFO のメンバー変数の説明**をご覧ください。

IKSetToClipBrd

【機能】
ラスタイメージやベクトルイメージをクリップボードへコピーします。

【関数書式】
(1)C++Builder
BOOL IKSetToClipBrd(HWND hWnd, HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKSetToClipBrd(hWnd: HWND; Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
hWnd	呼び出す側のウィンドウハンドル
Handle	ラスタイメージやベクトルイメージのメモリハンドル
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対象となるイメージがベクトルの場合、EMF 形式でクリップボードにコピーします。
引数として与えたメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Com.dll/Ik10ComA.dll/Ik10Com64.dll/Ik10Com64A.dll」のユーザ関数の定義**をご覧ください。

IKStretchBlt

【機能】

コピー元の矩形からコピー先の矩形へイメージをコピーします。このとき、必要に応じて、コピー先の矩形に合わせてイメージを拡大または縮小します。

【関数書式】

(1)C++Builder

BOOL IKStretchBlt(HDC hDCDst, long XDst, long YDst, long WidthDst, long HeightDst, HDC hDCSrc, long XSrc, long YSrc, long WidthSrc, long HeightSrc, DWORD dwRop);

(2)Delphi

function IKStretchBlt(hDCDst: HDC; XDst, YDst, WidthDst, HeightDst: Longint; hDCSrc: HDC; XSrc, YSrc, WidthSrc, HeightSrc: Longint; dwRop: DWORD): LongBool;

【引数】

名称	内容
hDCDst	コピー先のデバイスコンテキスト
XDst	コピー先矩形の左上隅の x 座標(ピクセル)
YDst	コピー先矩形の左上隅の y 座標(ピクセル)
WidthDst	コピー先矩形の幅(ピクセル)
HeightDst	コピー先矩形の高さ(ピクセル)
hDCSrc	コピー元のデバイスコンテキスト
XSrc	コピー元矩形の左上隅の x 座標(ピクセル)
YSrc	コピー元矩形の左上隅の y 座標(ピクセル)
WidthSrc	コピー元矩形の幅(ピクセル)
HeightSrc	コピー元矩形の高さ(ピクセル)
dwRop	ラスタオペレーションコード

【戻り値】

実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】

コピー元のイメージをメモリ内で拡大または縮小し、その結果をコピー先の矩形へコピーします。拡大または縮小を行った後で、パターンまたはコピー先ピクセルの色データを組み合わせます。

WidthSrc と WidthDest の各パラメータの符号、または HeightSrc と HeightDest の各パラメータの符号が異なる場合、ミラーイメージを作成します。WidthSrc と WidthDest の符号が異なる場合、x 軸を中心にしてミラーイメージを作成します。

HeightSrc と HeightDest の符号が異なる場合、y 軸を中心にしてミラーイメージを作成します。

ラスタオペレーションモードはデバイスコンテキストにどのようにコピーするかを指定します。

値	説明
0x00000042	コピー先の矩形を黒色で塗りつぶします。(BLACKNESS)
0x00550009	コピー先の矩形の色を反転します。(DSTINVERT)
0x00C000CA	論理 AND 演算子を使って、コピー元の色とコピー先の色を組み合わせます。(MERGECOPY)
0x00BB0226	論理 OR 演算子を使って、コピー元の色を反転した色と、コピー先の色を組み合わせます。(MERGEPAINT)
0x00330008	コピー元の色を反転して、コピー先へコピーします。(NOTSRCCOPY)
0x001100A6	論理 OR 演算子を使って、コピー元の色とコピー先の色を組み合わせ、さらに反転します。(NOTSRCERASE)
0x00F00021	指定したパターンをコピー先へコピーします。(PATCOPY)
0x005A0049	論理 XOR 演算子を使って、指定したパターンの色と、コピー先の色を組み合わせます。(PATINVERT)
0x00FB0A09	論理 OR 演算子を使って、指定したパターンの色と、コピー元の色を反転した色を組み合わせます。さらに論理 OR 演算子を使って、その結果と、コピー先の色を組み合わせます。(PATPAINT)
0x008800C6	論理 AND 演算子を使って、コピー元の色とコピー先の色を組み合わせます。(SRCAND)
0x00CC0020	コピー元の矩形をコピー先の矩形へそのままコピーします。(SRCCOPY)

0x00440328	論理 AND 演算子を使って、コピー先の色を反転した色と、コピー元の色を組み合わせます。(SRCERASE)
0x00660046	論理 XOR 演算子を使って、コピー元の色とコピー先の色を組み合わせます。(SRCINVERT)
0x00EE0086	論理 OR 演算子を使って、コピー元の色とコピー先の色を組み合わせます。(SRCPAINT)
0x00FF0062	コピー先の矩形を白色で塗りつぶします。(WHITENESS)

通常は 0x00CC0020(SRCCOPY)を指定します。

※16 進表記のため、Delphi は 0x を$に置き換えてください。
()内の説明は WindowsAPI で使用する定数と同じ意味です。

1－2. Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll

ラスタイメージに対して特殊処理を施す場合に使用するコマンドを提供します。

●RGB と YCrCb カラースペースについて
エフェクト処理ではしばし、RGB と YCrCb の変換が内部的に行われます。
以下に計算式を示します。
なお、Y は輝度情報、CrCb は色相情報を表します。

(1)RGB から YCrCb へ
Y = (0.29900 * R) + (0.58700 * G) + (0.11400 * B)
Cr = (0.50000 * R) + (−0.41869 * G) + (−0.08131 * B) + 128
Cb = (−0.16874 * R) + (−0.33126 * G) + (0.50000 * B) + 128

(2)YCrCb から RGB へ
R = Y + (1.402 * (Cr − 128))
G = Y − (0.34414 * (Cb − 128)) − (0.71414 * (Cr − 128))
B = Y + (1.772 * (Cb − 128))

●DLL 関数コマンド一覧（アルファベット順）

関数名	内容
IKAffine	アフィン変換処理
IKAntiAlias	アンチエリアシング処理
IKAutoSelectImageEx	自動選択処理
IKBlur	ぼかし処理
IKCanvas	キャンバス地効果
IKCheckSecretImage	透かしの取り出し関数
IKChroma	彩度補正処理
IKConvertColor	減色・増色処理
IKCustomFilter	カスタムフィルタ
IKCutRectImage	矩形切り出し処理
IKEmboss	エンボス処理
IKEndDibAccess	DIB アクセスの終了処理
IKGetDibPixel	DIB からピクセル値を取得
IKGlassTile	ガラスタイル効果
IKLayer(Ex)	ラスタイメージの重ね合わせ処理
IKLens	レンズ効果
IKMakeRGBAImage	RGB と A から RGBA の 32 ビットイメージを作成
IKMosaic	モザイク処理
IKMotionBlur	モーションぼかし処理
IKOilPaint	油絵風効果
IKOutline	輪郭抽出処理
IKPanorama	パノラマ処理
IKPasteImage	ラスタイメージの貼り付け処理
IKRedEyeRemoval	赤目補正
IKRemoveNoise	ノイズ除去処理
IKResizeEx	拡大縮小処理
IKRGBGamma	RGB 値のガンマ補正処理
IKRGBLevel	RGB 値の加減処理
IKRGBRev	RGB 値の反転処理
IKRGBSpline	RGB 値のスプライン補正処理
IKRipple	波紋効果
IKRotationEx	回転処理
IKSelectImageEx	自由選択処理
IKSetDibPixel	DIB へピクセル値を設定
IKSetGray	カラーイメージをグレースケールへ変換

IKSetSecretImage	透かしの埋め込み関数
IKSharp	シャープネス処理
IKSplitRGBAImage	RGBA の 32 ビットイメージを RGB(24 ビット)と A(8 ビット)に分割
IKStartDibAccess	DIB アクセスの開始処理
IKUnifyColor	色のばらつきを修正
IKWaves	さざ波効果
IKWhirlPinch	ねじりつまみ効果
IKYCCGamma	YCrCb 値のガンマ補正処理
IKYCCLevel	YCrCb 値の加減処理
IKYCCRev	YCrCb 値の反転処理
IKYCCSpline	YCrCb 値のスプライン補正処理

※**IK5** の **IK5Shade** 関数は **IK6** で **IKBlur** 関数に名称が変更されました(IK6 以降は同じ)。

●構造体(ユーザ定義型)の定義および解説

IKSELECT_IMAGE: 対象のイメージとマスクを表します。

(1)C++Builder

```
typedef struct {
    HANDLE    hImgBmh;
    HANDLE    hMskBmh;
} IKSELECT_IMAGE;
typedef IKSELECT_IMAGE * PTR_IKSELECT_IMAGE;
```

(2)Delphi

```
type
  IKSELECT_IMAGE = Record
    hImgBmh:  THandle;
    hMskBmh: THandle;
  end;
```

A.引数として扱う場合

hImgBmh:

処理対象となるイメージのハンドルを表します。(必ず設定する必要があります。)

hMskBmh:

hImgBmh に対するマスクイメージのハンドルを表します。(使用しない場合は 0 もしくはヌルを与えます。)
マスクイメージは 1 ビット(白黒 2 値)イメージでなければなりません。(それ以外のイメージを与えた場合は各関数でエラーが返されます。)
白色がマスクとして有効な部分になります。

hImgBmh と **hMskBmh** に設定するイメージは同じ幅と高さでなければなりません。

B.戻り値の場合

hImgBmh:

処理結果となるイメージのハンドルを表します。

HMskBmh:

処理結果となるマスクイメージのハンドルを表します。(引数に有効な値が設定された場合のみ値が返されます。)

IKDIB_INFO:　DIB にアクセスするために使用します。

(1)C++Builder
```
typedef struct {
    BYTE        Pal;
    BYTE        Red;
    BYTE        Green;
    BYTE        Blue;
    char        Reserved[128];
} IKDIB_INFO;
typedef IKDIB_INFO * PTR_IKDIB_INFO;
```

(2)Delphi
```
type
    IKDIB_INFO = Record
        Pal:        Byte;
        Red:        Byte;
        Green:      Byte;
        Blue:       Byte;
        Reserved:   array[0..127] of AnsiChar;
    end;
```

Pal:　パレット番号
Red:　RGB の赤
Green:　RGB の緑
Blue:　RGB の青
Reserved:　予約領域(ImageKit で使用するので、変更しないこと)

【ユーザ関数の定義】

【関数書式】
(1)C++Builder　BOOL _stdcall UserProc(short Percent);
(2)Delphi　function UserProc(Percent: Smallint): LongBool; stdcall;

【引数】

名称	内容
Percent	現在処理している%数

【戻り値】
False(0)の場合は実行している処理を終了します。True(0 以外)は処理を継続します。

ユーザ関数は UserProc という名称で説明しておりますが、実際の名称は何を設定されても構いません。関数名が UserProc の場合は以下のような形式で引数に渡します。
(1)C++Builder　UserProc
(2)Delphi　LONG_PTR(Addr(UserProc)) or LONG_PTR(@UserProc)

IKAffine

【機能】
歪んだラスタイメージを修正します。

【関数書式】
(1)C++Builder
HANDLE IKAffine(HANDLE Handle, int Px1, int Py1, int Px2, int Py2, int Px3, int Py3, int Px4, int Py4, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKAffine(Handle: THandle; Px1, Py1, Px2, Py2, Px3, Py3, Px4, Py4: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル(1,4,8,16,24,32 ビットイメージが対象となります)
Px1～Px4	アフィン変換する四点の X 座標(ピクセル)
Py1～Py4	アフィン変換する四点の Y 座標(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
(Px1,Py1)、(Px2,Py2)、(Px3,Py3)、(Px4,Py4)で囲まれたイメージをその点に内接する四角形に変換します。
引数として与えたメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKAntiAlias

【機能】
ラスタイメージのエッジを滑らかにします。

【関数書式】
(1)C++Builder
HANDLE IKAntiAlias(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKAntiAlias(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合は、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKAutoSelectImageEx

【機能】

指定した RGB 値に近似している部分を選択し、ラスタイメージとマスクを生成します。

【関数書式】

(1)C++Builder

BOOL IKAutoSelectImageEx(HANDLE Handle, PTR_IKSELECT_IMAGE DstHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE Red, BYTE Green, BYTE Blue, BYTE Mode, BYTE Level, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

(2)Delphi

function IKAutoSelectImageEx(Handle: THandle; var DstHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Red, Green, Blue, Mode, Level: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル(1,4,8,16,24,32 ビットイメージのいずれかを設定すること)
DstHandle	処理後のラスタイメージとマスクハンドルの構造体(各メンバーのイメージの大きさは同じ) (実行に失敗した場合はそれぞれのメンバーに、0 が返されます)
SelectMode	選択範囲の種類の指定(1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Red	選択対象となる赤の値(0～255)
Green	選択対象となる緑の値(0～255)
Blue	選択対象となる青の値(0～255)
Mode	比較モード(0:RGB, 1:CrCb, 2:Y)
Level	許容誤差(0～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】

実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】

イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。

lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。

SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。

Mode が 0 以外(CrCb:色相　Y:輝度)の場合には、設定された RGB 値を YCrCb に変換して比較します。

Level(許容誤差)は指定した RGB にどの程度近いピクセルを選択するかを制御します。値が 0 の場合は、指定した RGB 値そのものを選択し、値を大きくすると許容誤差が大きくなり、より多くのピクセルを選択します。引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKBlur

【機能】
ラスタイメージにぼかしを施します。

【関数書式】
(1)C++Builder
HANDLE IKBlur(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE Level, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKBlur(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Level: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Level	ぼかしの強さ(1～20　数が大きいほど強い)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
Level を大きくすればするほど、イメージはぼやけていきます。
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKCanvas

【機能】
ラスタイメージにキャンバス地効果を施します。

【関数書式】
(1)C++Builder
HANDLE IKCanvas(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE Direction, BYTE Depth, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKCanvas(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect: TRect; InOut: LongBool; Direction, Depth: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Direction	目地の方向(0:上から右、1:上から左、2:下から左、3:下から右)
Depth	目地の深さ(0～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
Depth が大きいほど彫りが深くなります。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKCheckSecretImage

【機能】
ラスタイメージから透かし情報を取り出します。

【関数書式】
(1)C++Builder
HANDLE IKCheckSecretImage(HANDLE Handle1, HANDLE Handle2, int Level, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKCheckSecretImage(Handle1, Handle2: THandle; Level: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle1	キーとなるラスタイメージのメモリハンドル(24 ビットイメージが対象)
Handle2	透かしが埋め込まれたラスタイメージのメモリハンドル(24 ビットイメージが対象)
Level	透かしの取り出しレベル(-20～20)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(1 ビットイメージ)。実行に失敗した場合は 0 が返されます

【解説】
IKSetSecretImage で設定した透かし情報を取得します。
Level には、**IKSetSecretImage** で設定した Level 付近、すなわち＋で埋め込んだ場合は埋め込み値より小さめな値を、－で埋め込んだ場合は埋め込み値より大きめな値を設定します。それ以外の値を設定した場合は、すかし情報が抽出されない場合があります。正常に処理が終了した場合の戻されるイメージは、黒の背景に白いすかし情報となります。(イメージの大きさは Handle1、Handle2 と同じ)
引数として与えたメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKChroma

【機能】
ラスタイメージの彩度を調整します。

【関数書式】
(1)C++Builder
HANDLE IKChroma(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, int Level, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKChroma(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Level: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Level	彩度の加減 (−100～1000)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Level が 0 の場合には何も行わず、-100 の場合は見かけ上グレースケールと同じになります。
－に設定すると色は相対的に淡くなり、＋に設定すると鮮やかになります。

マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
1,4,8 ビットイメージの場合は、SelectMode が 0～3 のいずれが設定されていてもイメージ全体に対して処理を行います。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKConvertColor

【機能】
メモリ上のラスタイメージの色数を変更します。

【関数書式】
(1)C++Builder
HANDLE IKConvertColor(HANDLE Handle, short PixelType, BOOL FixedPal, BOOL Dither, BYTE Level, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKConvertColor(Handle: THandle; PixelType: Smallint; FixedPal, Dither: LongBool; Level: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル(1,4,8,16,24,32 ビットイメージ)
PixelType	変更後のピクセル当たりのビット数(1,4,8,16,24,32,40,80) ※4 は 4 ビットカラー、40 は 4 ビットグレー、8 は 8 ビットカラー、80 は 8 ビットグレーとなります。
FixedPal	固定パレットの使用設定(1,4,8 ビットカラーに減色および増色する際に使用) False(0):イメージに合わせたパレットを作成し使用 True(0 以外):固定パレットを使用
Dither	ディザリング(誤差拡散法)の設定(1,4,8 ビットカラーおよび 4 ビットグレーに減色する際に使用) False(0):ディザリングを行わない True(0 以外):ディザリングを行う
Level	しきい値の設定(1 ビットイメージに減色する際に使用) 0～255
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
色数変更後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
1 ビットイメージに減色した場合に、ピクセルの RGB の輝度が Level で設定された値以上の場合はパレット 1 を、それ未満の場合はパレット 0 として処理を行います。
引数として与えたメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKCustomFilter

【機能】
ラスタイメージに対してユーザ独自のフィルタ処理を施します。

【関数書式】
(1)C++Builder
HANDLE IKCustomFilter(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, LPINT Matrix, BYTE Div, BYTE Level, BOOL Red, BOOL Green, BOOL Blue, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKCustomFilter(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; var Matrix: Integer; Div, Level: Byte; Red, Green, Blue: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Matrix	9×9 行列の係数(配列、項目数は 81 で添字は 0～80) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Mt: array [0..80] of Integer 引数に Mt[0]を与える
Div	行列に対する除数
Level	行列に対する加減(0～255 大きいほど明るい)
Red	赤のプレーンに対する処理(False(0):しない、True(0 以外):する)
Green	緑のプレーンに対する処理(False(0):しない、True(0 以外):する)
Blue	青のプレーンに対する処理(False(0):しない、True(0 以外):する)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
グレースケールのイメージを処理する場合は、Red、Green、Blue で設定された値は無効となります。

引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

出力するピクセルの計算式は以下のようになります。

$$f = \left\{ \sum_{i=0}^{80} PiCi \right\} \div Div + Level$$

Pi はピクセルの値、Ci は Matrix の係数の値を表します。

簡単なエンボスフィルタの例を示します。

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	−1	0	1	0	0	0
0	0	0	−1	0	1	0	0	0
0	0	0	−1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Matrix[0]～Matrix[8]
Matrix[9]～Matrix[17]
Matrix[18]～Matrix[26]
Matrix[27]～Matrix[35]
Matrix[36]～Matrix[44]
Matrix[45]～Matrix[53]
Matrix[54]～Matrix[62]
Matrix[63]～Matrix[71]
Matrix[72]～Matrix[80]

Div = 1、Level = 128

上の表の値は Matrix の配列のデータを示します。(左から右へ、上から下へいく毎に添字が増える)

IKCutRectImage

【機能】
ラスタイメージから指定した矩形領域を切り出します。

【関数書式】
(1)C++Builder
HANDLE IKCutRectImage(HANDLE Handle, LPRECT Rect, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKCutRectImage(Handle: THandle; var Rect: TRect; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル(1,4,8,16,24,32 ビットイメージが対象)
Rect	矩形の座標を指定する構造体変数
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
マスクイメージを必要としない矩形の切り出しに使用できます。(マスクイメージが必要な場合は **IKSelectImageEx** 関数をご使用ください。)
処理速度は **IKSelectImageEx** と比較して高速です。
引数として与えた Handle のメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKEmboss

【機能】
ラスタイメージにエンボス(浮き彫り)処理を施します。

【関数書式】
(1)C++Builder
HANDLE IKEmboss(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE MskPattern, BYTE Edge, BYTE Level, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKEmboss(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; MskPattern, Edge, Level: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
MskPattern	フィルタマスクのパターン(0～7)
Edge	エッジの強調(1～5　大きいほど強い)
Level	背景の輝度(0～255　大きいほど明るい)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
MskPattern によって浮き彫り方向が変わります。

マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKEndDibAccess

【機能】
DIB へのアクセス処理を終了します。

【関数書式】
(1)C++Builder
BOOL IKEndDibAccess(HANDLE hDib, PTR_IKDIB_INFO stpDibInfo);
(2)Delphi
function IKEndDibAccess(hDib: THandle; var stpDibInfo: IKDIB_INFO): LongBool;

【引数】

名称	内容
hDib	ラスタイメージのメモリハンドル(1,4,8,16,24,32 ビットイメージ)
stpDibInfo	**IKStartDibAccess** 関数で取得した構造体(ユーザ定義型)変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
IKStartDibAccess と対で使用します。
hDib と stpDibInfo は IKStartDibAccess で使用した引数と同じでなければなりません。
IKDIB_INFO については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」の IKDIB_INFO のメンバー変数の説明**をご覧ください。

処理の流れとしては
IKStartDibAccess
|
IKGetDibPixel, IKSetDibPixel
|
IKEndDibAccess
となります。

コード例については IKStartDibAccess を参照してください。

IKGetDibPixel

【機能】
DIB から指定したピクセルの RGB やパレット番号を取得します。

【関数書式】
(1)C++Builder
BOOL IKGetDibPixel(PTR_IKDIB_INFO stpDibInfo, long lx, long ly);
(2)Delphi
function IKGetDibPixel(var stpDibInfo: IKDIB_INFO: lx, ly: Longint): LongBool;

【引数】

名称	内容
stpDibInfo	**IKStartDibAccess** 関数で取得した構造体(ユーザ定義型)変数
lx, ly	取得するピクセルの X,Y 座標

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
成功した場合、1,4,8 ビットイメージの場合は stpDibInfo の Pal に、16,24,32 ビットイメージの場合は stpDibInfo の Red,Green,Blue に有効な値が設定されます。
IKDIB_INFO については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」の IKDIB_INFO のメンバー変数の説明**をご覧ください。

処理の流れとしては
IKStartDibAccess
　　|
IKGetDibPixel, IKSetDibPixel
　　|
IKEndDibAccess
となります。

コード例については IKStartDibAccess を参照してください。

IKGlassTile

【機能】
ラスタイメージにガラスタイル効果を施します。

【関数書式】
(1)C++Builder
HANDLE IKGlassTile(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, int xSize, int ySize, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKGlassTile(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect: TRect; InOut: LongBool; xSize, ySize: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
xSize	ガラスタイルの横サイズ(10～100)
ySize	ガラスタイルの縦サイズ(10～100)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKLayer,IKLayerEx

【機能】
ラスタイメージの重ね合わせを行います。

【関数書式】
(1)C++Builder
HANDLE **IKLayer**(HANDLE Handle1, HANDLE Handle2, BYTE Trans, BOOL TransColor, BYTE TRed, BYTE TGreen, BYTE TBlue, BYTE BRed, BYTE BGreen, BYTE BBlue, int x, int y, BOOL Clip, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
HANDLE **IKLayerEx**(HANDLE Handle1, HANDLE Handle2, BYTE Trans, BOOL TransColor, BYTE TRed, BYTE TGreen, BYTE TBlue, BYTE BRed, BYTE BGreen, BYTE BBlue, int x, int y, BOOL Clip, BOOL Alpha, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function **IKLayer**(Handle1, Handle2: THandle; Trans: Byte; TransColor: LongBool; TRed, TGreen, TBlue, BRed, BGreen, BBlue: Byte; x, y: Integer; Clip: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;
function **IKLayerEx**(Handle1, Handle2: THandle; Trans: Byte; TransColor: LongBool; TRed, TGreen, TBlue, BRed, BGreen, BBlue: Byte; x, y: Integer; Clip, Alpha: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle1	重ねられるラスタイメージのメモリハンドル(1,4,8,16,24,32 ビットイメージ)
Handle2	重ねるラスタイメージのメモリハンドル(1,4,8,16,24,32 ビットイメージ)
Trans	透かしの度合い(0～255 数が大きいほど Handle2 が生きる)
TransColor	Handle2 に対する透明色の設定(False(0):なし、True(0 以外):あり)
TRed	Handle2 に対する透明色 赤(0～255)
TGreen	Handle2 に対する透明色 緑(0～255)
TBlue	Handle2 に対する透明色 青(0～255)
BRed	重ね合わせたイメージの無効領域の背景色 赤(0～255)
BGreen	重ね合わせたイメージの無効領域の背景色 緑(0～255)
BBlue	重ね合わせたイメージの無効領域の背景色 青(0～255)
x,y	貼り付け座標(ピクセル)
Clip	クリッピングの有無(False(0):なし、True(0 以外):あり)
Alpha	アルファチャネルを考慮(False(0):しない、True(0 以外):する) ※IKLayerEx で使用
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドルが返されます。(実行に失敗した場合は 0)

出力されるイメージのビット数の説明
- Handle1とHandle2のビット数が共に8ビット以下で、かつ同じビット数であり、かつ同じパレット数と並びの場合は入力イメージのビット数と同じになります。ただし、Handle1 と Handle2 が 8 ビットグレースケールではなく Trans が 0 か 255 以外の場合は 24 ビットイメージになります。
- Handle1とHandle2のビット数が共に8ビット以下でも、ビット数が違うか、ビット数が同じでもパレット数と並びが違う場合は 24 ビットイメージになります。
- Handle1 と Handle2 のどちらかが 16 ビット以上の場合は、大きいビット数になります。

【解説】
Handle2 のイメージの左上の点を、Handle1 のイメージの x,y の位置に合わせて貼り付けを行います。
Clip が True の時は、Handle1 のイメージのサイズより大きくなった場合に、はみでた領域をカットします。
TransColor が True(0 以外)の場合は、TRed・TGreen・TBlue で設定された色を透明色として使用します。

アルファチャンネルを考慮して重ね合わせをする場合は、**IKLayerEx** 関数で Alpha を True(0 以外)に設定して、かつ Handle2 に有効なアルファチャンネルが入った RGBA の 32 ビットイメージを与える必要があります。
Alpha が True(0 以外)の場合は Trans は無効になり、False(0)の場合は **IKLayer** 関数と同じ動作となります。
引数として与えたメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKLayerEx 関数の使用例としては、ある JPEG 画像の上にアルファチャンネルの設定のある PNG 画像を重ね合わせる場合が考えられます。

【ImageKit5 との違い】

関数名	**引数の並び**
IK5Layer	Handle1, Handle2, Trans, TRed, TGreen, TBlue, BRed, BGreen, BBlue, x, y, Clip, EffectUserProc, Caption, Message, Button
IKLayer	Handle1, Handle2, Trans, TransColor, TRed, TGreen, TBlue, BRed, BGreen, BBlue, x, y, Clip, UserProc, Caption, Message, Button

IK6 から引数に TransColor が追加されています。TransColor を True(0 以外)に設定すると IK5 と同じ動作となります。
※EffectUserProc と UserProc は同じユーザ関数を表します。

IKLens

【機能】
ラスタイメージにレンズ効果を施します。

【関数書式】
(1)C++Builder
HANDLE IKLens(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, double Refract, BOOL BackColor, BYTE Red, BYTE Green, BYTE Blue, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKLens(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect: TRect; InOut: LongBool; Refract: Double; BackColor: LongBool; Red, Green, Blue: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Refract	レンズの倍率(1.0 以上)
BackColor	レンズの外側の背景色の描画設定 [True(0 以外):塗る　False(0):塗らない]
Red	レンズの外側の背景色 赤(0～255)
Green	レンズの外側の背景色 緑(0～255)
Blue	レンズの外側の背景色 青(0～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
BackColor が True(0 以外)の時、Red,Green,Blue が有効になります。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKMakeRGBAImage

【機能】
RGB と A から RGBA の 32 ビットイメージを作成します。

【関数書式】
(1)C++Builder
HANDLE IKMakeRGBAImage(HANDLE InHandle1, HANDLE InHandle2, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKMakeRGBAImage(InHandle1, InHandle2: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
InHandle1	RGB イメージのメモリハンドル(1,4,8,16,24,32 ビットイメージが対象)
InHandle2	A(アルファチャンネル)イメージのメモリハンドル(1,8 ビットグレースケールイメージが対象)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
RGBA の 32 ビットイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

【参照】
「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」の IKPngFileLoadEx(Mem),IKPngFileSaveEx(Mem)関数

IKMosaic

【機能】
ラスタイメージにモザイク処理を施します。

【関数書式】
(1)C++Builder
HANDLE IKMosaic(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKMosaic(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Width	モザイクの幅(ピクセル)
Height	モザイクの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Width と Height が共に 1 以下の場合はモザイク処理を行いません。

マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKMotionBlur

【機能】
ラスタイメージにモーションぼかし効果を施します。

【関数書式】
(1)C++Builder
HANDLE IKMotionBlur(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE Mode, BYTE Length, int Angle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKMotionBlur(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect: TRect; InOut: LongBool; Mode, Length: Byte; Angle: Inetegr; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Mode	ぼかしの種類(0:線形, 1:放射状, 2:拡大)
Length	ぼかしの度合い(長さ、0～255)
Angle	ぼかしの度合い(角度、1 度単位)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKOilPaint

【機能】
ラスタイメージに油絵風効果を施します。

【関数書式】
(1)C++Builder
HANDLE IKOilPaint(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE Mode, BYTE MaskSize, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKOilPaint(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect: TRect; InOut: LongBool; Mode, MaskSize: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Mode	油絵の質感モード(0:RGB 別処理, 1:輝度別処理)
MaskSize	筆のサイズ (1～100)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKOutline

【機能】
ラスタイメージから輪郭部分を抽出します。

【関数書式】
(1)C++Builder
HANDLE IKOutline(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE Plane, BYTE Operator, BYTE Level, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKOutline(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Plane, Operator, Level: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Plane	処理するプレーン(0:Y,1:R,2:G,3:B)
Operator	行列変数のパターン(0:差分, 1:Roberts, 2:Sobel)
Level	輪郭線の明るさ(数が大きいほど明るい　0～100)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
通常は Plane を 0 として輝度の違いを輪郭として認識しますが、1 以上を設定することにより各プレーンの値の差を輪郭として認識することもできます。輪郭抽出の効果は Operator により異なります。
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKPanorama

【機能】
ラスタイメージのパノラマ合成を行います。

【関数書式】
(1)C++Builder
HANDLE IKPanorama(HANDLE Handle1, HANDLE Handle2, BYTE CutMode, int Px11, int Py11, int Px12, int Py12, int Px21, int Py21, int Px22, int Py22, BYTE Red, BYTE Green, BYTE Blue, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKPanorama(Handle1, Handle2: THandle; CutMode: Byte; Px11, Py11, Px12, Py12, Px21, Py21, Px22, Py22: Integer; Red, Green, Blue: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle1	基となるラスタイメージのハンドル(24 ビットイメージ)
Handle2	貼り付けるラスタイメージのハンドル(24 ビットイメージ)
CutMode	Handle2 の基準点に対してカットする方向を設定(0:なし 1:左 2:右)
Px11,Py11	Handle1 のイメージ上の基準点 1(ピクセル)
Px12,Py12	Handle1 のイメージ上の基準点 2(ピクセル)
Px21,Py21	Handle2 のイメージ上の基準点 1(ピクセル)
Px22,Py22	Handle2 のイメージ上の基準点 2(ピクセル)
Red	重ね合わせたイメージの無効領域の背景色 赤(0～255)
Green	重ね合わせたイメージの無効領域の背景色 緑(0～255)
Blue	重ね合わせたイメージの無効領域の背景色 青(0～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(24 ビットイメージ)。実行に失敗した場合は 0 が返されます。

【解説】
Handle1 の Px11、Py11、Px12、Py12 で指定した座標に、Handle2 の Px21、Py21、Px22、Py22 で指定した座標を合わせて、CutMode の値により Handle2 のイメージをそのまま合成するかどうかを設定し、Handle1 に Handle2 のイメージを合成します。(Handle2 の基準点が Handle1 の基準点に合わさるような形になります。)
CutMode は基準点 1 を上に基準点 2 を下にした場合の、カットする方向を示しています。
引数として与えたメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義をご覧ください。

IKPasteImage

【機能】
ラスタイメージの貼り付けを行います。

【関数書式】
(1)C++Builder
HANDLE IKPasteImage(HANDLE Handle, PTR_IKSELECT_IMAGE SrcHandle, int Angle, BOOL TurnX, BOOL TurnY, BYTE Trans, BOOL TransColor, BYTE TRed, BYTE TGreen, BYTE TBlue, BYTE BRed, BYTE BGreen, BYTE BBlue, int x, int y, BOOL Clip, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKPasteImage(Handle: THandle; var SrcHandle: IKSELECT_IMAGE; Angle: Integer; TurnX, TurnY: LongBool; Trans: Byte; TransColor: LongBool; TRed, TGreen, TBlue, BRed, BGreen, BBlue: Byte; x, y: Integer; Clip: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	貼り付けられるラスタイメージのハンドル(1,4,8,16,24,32 ビットイメージ)
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
Angle	回転角度(-35999～35999　1/100 度単位)
TurnX	X 方向反転の有無 (False(0):反転なし True(0 以外):反転あり)
TurnY	Y 方向反転の有無 (False(0):反転なし True(0 以外):反転あり)
Trans	透かしの度合い (0～255 数が大きいほど SrcHandle が生きる)
TransColor	SrcHandle(**hImgBmh**)に対する透明色の設定(False(0):なし、True(0 以外):あり)
TRed	SrcHandle(**hImgBmh**)に対する透明色 赤(0～255)
TGreen	SrcHandle(**hImgBmh**)に対する透明色 緑(0～255)
TBlue	SrcHandle(**hImgBmh**)に対する透明色 青(0～255)
BRed	貼り付けたイメージの無効領域の背景色 赤(0～255)
BGreen	貼り付けたイメージの無効領域の背景色 緑(0～255)
BBlue	貼り付けたイメージの無効領域の背景色 青(0～255)
x,y	貼り付け座標(ピクセル)
Clip	クリッピングの有無(False(0):なし、True(0 以外):あり)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

出力されるイメージのビット数の説明
- Handle と SrcHandle のビット数が共に 8 ビット以下で、かつ同じビット数であり、かつ同じパレット数と並びの場合は入力イメージのビット数と同じになります。ただし、Handle と SrcHandle が 8 ビットグレースケールではなく Trans が 0 か 255 以外の場合は 24 ビットイメージになります。
- Handle と SrcHandle のビット数が共に 8 ビット以下でも、ビット数が違うか、ビット数が同じでもパレット数と並びが違う場合は 24 ビットイメージになります。
- Handle と SrcHandle のどちらかが 16 ビット以上の場合は、大きいビット数になります。

【解説】
Angle、TurnX、TurnY は SrcHandle に対して有効です。
x,y は Handle の左上座標(0,0)として、SrcHandle の指すイメージの中心点を Handle のどの座標に貼り付けるかを指定します。Clip が True の時は、Handle のイメージのサイズより大きくなった場合に、はみでた領域をカットします。
TransColor が True(0 以外)の場合は、TRed・TGreen・TBlue で設定された色を透明色として使用します。

引数として与えたメモリハンドル(構造体も含む)は解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義をご覧ください。

【ImageKit5 との違い】

関数名	引数の並び
IK5PasteImage	Handle, SrcHandle, Angle, TurnX, TurnY, Trans, Red, Green, Blue, x, y, Clip, EffectUserProc, Caption, Message, Button
IKPasteImage	Handle, SrcHandle, Angle, TurnX, TurnY, Trans, TransColor, TRed, TGreen, TBlue, BRed, BGreen, BBlue, x, y, Clip, UserProc, Caption, Message, Button

IK6 から引数に TransColor,TRed,TGreen,TBlue が追加され、IK5 で Red,Green,Blue だったものが、IK6 から BRed,BGreen,BBlue に変更されています。
TransColor を False(0)に設定すると IK5 と同じ動作となります。
※EffectUserProc と UserProc は同じユーザ関数を表します。

IKRedEyeRemoval

【機能】
赤目を指定した色に補正します。

【関数書式】
(1)C++Builder
HANDLE IKRedEyeRemoval(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE Red, BYTE Green, BYTE Blue, BYTE Error, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKRedEyeRemoval(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Red, Green, Blue, Error: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Red	補正後の赤(0～255)
Green	補正後の緑(0～255)
Blue	補正後の青(0～255)
Error	補正する赤目の誤差範囲(0～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Error は値が大きい程より赤に近い色を補正し、小さい程補正する色の範囲が広がります。
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKRemoveNoise

【機能】
ラスタイメージのノイズを除去します。

【関数書式】
(1)C++Builder
HANDLE IKRemoveNoise(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE Mode, BYTE Level, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKRemoveNoise(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Mode, Level: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Mode	処理モード(0:輝度値のメディアン法,1:RGB 別のノイズ除去)
Level	RGB でのノイズ除去レベル(0～255、Mode が 1 の時に有効)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Mode が 1 の場合には Level で設定された値が有効になります。Level を高く設定するほどノイズを除去することができます。

マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

【ImageKit8 との違い】
　白黒 2 値画像に対応しました（引数の SrcHandle に白黒 2 値画像を渡すことができる）。

IKResizeEx

【機能】
ラスタイメージのサイズを変更します。

【関数書式】
(1)C++Builder
BOOL IKResizeEx(PTR_IKSELECT_IMAGE SrcHandle, PTR_IKSELECT_IMAGE DstHandle, long Width, long Height, BOOL Mode, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKResizeEx(var SrcHandle, DstHandle: IKSELECT_IMAGE; Width, Height: Longint; Mode: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
DstHandle	処理後のラスタイメージとマスクハンドルの構造体(各メンバーのイメージの大きさは同じ) (実行に失敗した場合はそれぞれのメンバーに、0 が返されます)
Width	処理後のイメージの幅(ピクセル)
Height	処理後のイメージの高さ(ピクセル)
Mode	補間の有無(False(0):なし、True(0 以外):あり) True の場合、拡大時は線形補間を行い、縮小時は平均を取り出力 8 ビットグレースケール,16,24,32 ビットイメージが対象となり、それ以外は無視される
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
基となるイメージを、Width・Height で指定した新しいサイズになるように自動的に縮小もしくは拡大してサイズを変更します。
Mode が False の場合、基のイメージより大きくするとその拡大率に比例してイメージが粗くなります。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKRGBGamma

【機能】
ラスタイメージの RGB 値のガンマ補正を行います。

【関数書式】
(1)C++Builder
HANDLE IKRGBGamma(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, double Red, double Green, double Blue, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKRGBGamma(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Red, Green, Blue: Double; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Red	Red のガンマ係数
Green	Green のガンマ係数
Blue	Blue のガンマ係数
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
処理後のピクセル値を y、基のピクセル値を x とすると
(1)Bright ＞ 0 の場合

$$y = (x/255)^{Bright+1} \times 255$$

(2)Bright ＜ 0 の場合

$$y = (x/255)^{-1/Bright-1} \times 255$$

となります。

Bright は Red、Green、Blue のいずれかになります。Red、Green、Blue が 0 の場合は処理の前後で変化ありません。Red、Green、Blue が＋の場合はレベルが下がり、－の場合はレベルが上がります。

マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、

SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
1,4,8 ビットイメージの場合は、SelectMode が 0～3 のいずれが設定されていてもイメージ全体に対して処理を行います。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKRGBLevel

【機能】
ラスタイメージの RGB 値の加減処理を行います。

【関数書式】
(1)C++Builder
HANDLE IKRGBLevel(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, int Red, int Green, int Blue, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKRGBLevel(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Red, Green, Blue: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Red	Red の加減(-255～255)
Green	Green の加減(-255～255)
Blue	Blue の加減(-255～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
Red,Green,Blue のそれぞれの値が大きくなると該当する成分が明るくなり、小さくなると該当する成分が暗くなります。また、Red,Green,Blue が 0 の場合は処理の前後で変化ありません。
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
1,4,8 ビットイメージの場合は、SelectMode が 0～3 のいずれが設定されていてもイメージ全体に対して処理を行います。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKRGBRev

【機能】
ラスタイメージの RGB 値の反転処理を行います。

【関数書式】
(1)C++Builder
HANDLE IKRGBRev(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BOOL Red, BOOL Green, BOOL Blue, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKRGBRev(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Red, Green, Blue: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Red	Red の反転(False(0): 反転なし, True(0 以外): 反転あり)
Green	Green の反転(False(0): 反転なし, True(0 以外): 反転あり)
Blue	Blue の反転(False(0): 反転なし, True(0 以外): 反転あり)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
Red、Green、Blue を True にするとそれぞれの RGB 値を反転します。
例えば、Red が 255 の場合は 0 となります。(Red = 255 - Red)
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
1,4,8 ビットイメージの場合は、SelectMode が 0～3 のいずれが設定されていてもイメージ全体に対して処理を行います。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKRGBSpline

【機能】
ラスタイメージの RGB 値のスプライン補正を行います。

【関数書式】
(1)C++Builder
HANDLE IKRGBSpline(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, LPPOINT lpSpPoint, int SpPoints, BOOL Red, BOOL Green, BOOL Blue, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKRGBSpline(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; var lpSpPoint: TPoint; SpPoints: Integer; Red, Green, Blue: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

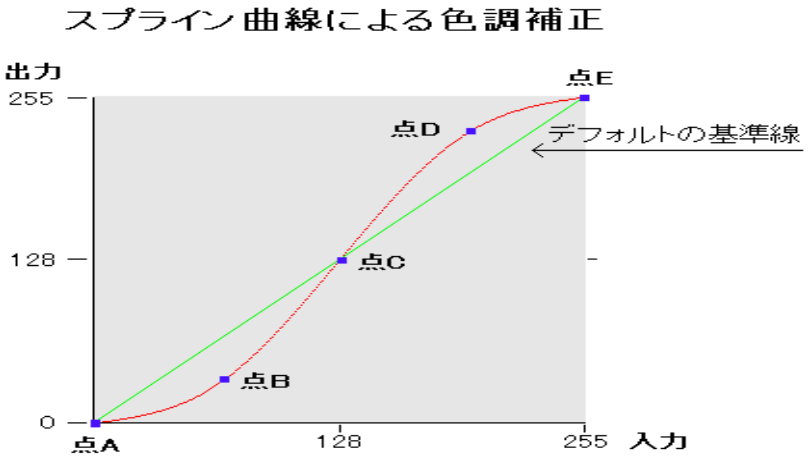
名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
lpSpPoint	スプラインの通過座標を指定する構造体変数の配列(値の設定方法は、lpPoint と同じ)
SpPoints	lpSpPoint の配列の数(3～10 が有効)
Red	赤の成分に対して処理を行うかどうかの設定(False(0):しない　True(0 以外):する)
Green	緑の成分に対して処理を行うかどうかの設定(False(0):しない　True(0 以外):する)
Blue	青の成分に対して処理を行うかどうかの設定(False(0):しない　True(0 以外):する)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
入力されたイメージデータの RGB 値に対してスプライン曲線により変化させることができます。

「使用例」
lpSpPoint の値　点 A(0,0)　点 B(64,30)　点 C(128,128)　点 D(192,225)　点 E(255,255)
SpPoints の値　5

・作成される曲線は lpSpPoint を通過するスプライン曲線を描きます。(最大 10 ポイントまで)



マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
1,4,8 ビットイメージの場合は、SelectMode が 0～3 のいずれが設定されていてもイメージ全体に対して処理を行います。
lpSpPoint のメンバーには 0～255 の間の値を設定してください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKRipple

【機能】
ラスタイメージに波紋効果を施します。

【関数書式】
(1)C++Builder
HANDLE IKRipple(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, double Amp, double Wavelen, double Phase, BOOL BackColor, BOOL Reflect, BYTE Red, BYTE Green, BYTE Blue, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKRipple(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Amp, Wavelen, Phase: Double; BackColor, Reflect: LongBool; Red, Green, Blue: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Amp	波の振幅の大きさ
Wavelen	波長の長さ
Phase	波の位相
BackColor	空白部分の設定 [False(0):背景色で塗る,True(0 以外):隣接するピクセルで塗る]
Reflect	波の反射設定 [False(0):しない,True(0 以外):する]
Red	背景色の赤(0～255)
Green	背景色の緑(0～255)
Blue	背景色の青(0～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。
その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
Red,Green,Blue は BackColor が False(0)の場合に有効です。

引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義をご覧ください。

IKRotationEx

【機能】
ラスタイメージを回転させます。

【関数書式】
(1)C++Builder
BOOL IKRotationEx(PTR_IKSELECT_IMAGE SrcHandle, PTR_IKSELECT_IMAGE DstHandle, int Angle, BOOL TurnX, BOOL TurnY, BOOL Mode, BYTE Red, BYTE Green, BYTE Blue, BOOL Clip, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKRotationEx(var SrcHandle, DstHandle: IKSELECT_IMAGE; Angle: Integer; TurnX, TurnY, Mode: LongBool; Red, Green, Blue: Byte; Clip: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): LonBool;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
DstHandle	処理後のラスタイメージとマスクハンドルの構造体(各メンバーのイメージの大きさは同じ) (実行に失敗した場合はそれぞれのメンバーに 0 が返されます)
Angle	回転角度 (-35999～35999　1/100 度単位)
TurnX	X 方向反転の有無 (False(0):反転なし True(0 以外):反転あり)
TurnY	Y 方向反転の有無 (False(0):反転なし True(0 以外):反転あり)
Mode	補間の有無(False(0):なし、True(0 以外):あり) True の場合は線形補間を行う 8 ビットグレースケール,16,24,32 ビットイメージが補間の対象となる
Red	イメージの背景色の赤 (0～255)
Green	イメージの背景色の緑 (0～255)
Blue	イメージの背景色の青 (0～255)
Clip	クリッピングの有無(False(0):なし、True(0 以外):あり)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
Angle の値が＋の場合は反時計周りに回転し、－の場合は時計回りに回転します。
Clip が True の時は、基のイメージのサイズより大きくなった場合に、はみでた領域をカットします。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKSelectImageEx

【機能】
選択範囲を基にラスタイメージとマスクイメージの部分生成を行います。

【関数書式】
(1)C++Builder
BOOL IKSelectImageEx(HANDLE Handle, PTR_IKSELECT_IMAGE DstHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE Red, BYTE Green, BYTE Blue, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKSelectImageEx(Handle: THandle; var DstHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Red, Green, Blue: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
Handle	ラスタイメージのハンドル(1,4,8,16,24,32 ビットイメージ)
DstHandle	処理後のラスタイメージとマスクハンドルの構造体(各メンバーのイメージの大きさは同じ) (実行に失敗した場合はそれぞれのメンバーに 0 が返されます)
SelectMode	選択範囲の種類の指定(1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Red	選択外領域の赤の値(0～255)
Green	選択外領域の緑の値(0～255)
Blue	選択外領域の青の値(0～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。

選択した領域を含む最小限な矩形を生成して、IKSELECT_IMAGE の構造体のメンバーにそれぞれのイメージハンドルをセットします。
戻り値としてセットされるラスタイメージ(マスクイメージでない)の選択した領域外は、Red、Green、Blue で指定された RGB 値により塗り潰されます。マスクイメージは、選択した領域が白で塗り潰されます。(領域外は黒)
引数として与えたメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。
矩形の切り出しでマスクイメージが不要な場合は、**IKCutRectImage** 関数を使用することにより処理速度が向上します。

IKSetDibPixel

【機能】
指定したピクセルの RGB やパレット番号を DIB に設定します。

【関数書式】
(1)C++Builder
BOOL IKSetDibPixel(PTR_IKDIB_INFO stpDibInfo, long lx, long ly);
(2)Delphi
function IKSetDibPixel(var stpDibInfo: IKDIB_INFO: lx, ly: Longint): LongBool;

【引数】

名称	内容
stpDibInfo	**IKStartDibAccess** 関数で取得した構造体(ユーザ定義型)変数
lx, ly	設定するピクセルの X,Y 座標

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
1,4,8 ビットイメージの場合は stpDibInfo の PalNo に、16,24,32 ビットイメージの場合は stpDibInfo の Red,Green,Blue に有効な値を設定します。
IKDIB_INFO については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」の IKDIB_INFO のメンバー変数の説明**をご覧ください。

処理の流れとしては
IKStartDibAccess
|
IKGetDibPixel, IKSetDibPixel
|
IKEndDibAccess
となります。

コード例については IKStartDibAccess を参照してください。

IKSetGray

【機能】
ラスタイメージをグレースケールに変換します。

【関数書式】
(1)C++Builder
HANDLE IKSetGray(HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKSetGray(Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル(1,4,8,16,24,32 ビットイメージ)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
引数として与えたメモリハンドルは解放されずにそのまま残ります。
IKSetGray 関数はビット数をそのままに処理を行いますので、8 ビットグレースケールに変換する場合は **IKConvertColor** 関数をご使用ください。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKSetSecretImage

【機能】
透かし情報をラスタイメージに埋め込みます。

【関数書式】
(1)C++Builder
HANDLE IKSetSecretImage(HANDLE Handle1, HANDLE Handle2, LPCTSTR Text, LPCTSTR FName, WORD FSize, BOOL Bold, BOOL Italic, BOOL Underline, BOOL StrikeOut, BYTE Direction, int Angle, int Level, int Left, int Top, int Right, int Bottom, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKSetSecretImage(Handle1, Handle2: THandle; Text, FName: PChar; FSize: Word; Bold, Italic, Underline, StrikeOut: LongBool; Direction: Byte; Angle, Level, Left, Top, Right, Bottom: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle1	基となるラスタイメージのハンドル(24 ビットイメージ)
Handle2	埋め込むラスタイメージのハンドル(1 ビットイメージ)
Text	埋め込む文字列(Ansi 版はヌルを含めて 128 バイト以下、Unicode 版はヌルを含めて 128 文字以下)
FName	フォントの名称(Ansi 版はヌルを含めて 32 バイト以下、Unicode 版はヌルを含めて 32 文字以下)
FSize	フォントのサイズ(ポイントで指定)
Bold	False(0)以外のとき、ボールド体のフォント
Italic	False(0)以外のとき、イタリック体のフォント
Underline	False(0)以外のとき、下線付きのフォント
StrikeOut	False(0)以外のとき、打ち消し線付きのフォント
Direction	文字列の方向(0:左から右 1:右から左 2:上から下 3:下から上)
Angle	文字の回転角度(0, 90, 180, 270)
Level	埋め込む輝度のレベル(-20～20)
Left	Handle1 のイメージに対して文字列または Handle2 のイメージを埋め込む左上の X 座標
Top	Handle1 のイメージに対して文字列または Handle2 のイメージを埋め込む左上の Y 座標
Right	Handle1 のイメージに対して文字列を埋め込む右下の X 座標
Bottom	Handle1 のイメージに対して文字列を埋め込む右下の Y 座標
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(24 ビットイメージ)。実行に失敗した場合は 0 が返されます。

【解説】
Level を大きくすると透かし情報を埋め込んだところが明るくなり、小さくすると透かし情報を埋め込んだところが暗くなります。
Underline と StrikeOut は、False(0)以外でも Direction や Angle の値により無効となる場合があります。
Left、Top、Right、Bottom はピクセル単位で設定します。
Handle2 が 0 でない場合は、Handle2 のイメージが埋め込む透かし情報となります。
Handle2 が 0 の場合は、Text で指定した文字列が透かし情報となり、Left、Top、Right、Bottom で指定した範囲に収まるようにセットします。

128 バイト以上の文字情報をセットする、もしくは複数の場所に透かし情報を埋め込みたい場合は、Handle2 のイメージをそのように作成して対応してください。
引数として与えたメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKSharp

【機能】
ラスタイメージの輪郭を強調してシャープにします。

【関数書式】
(1)C++Builder
HANDLE IKSharp(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE Level, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKSharp(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Level: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Level	シャープネスの強さ(0～100　数が大きいほど強い)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Level を大きくすればするほど、イメージはくっきりしていきます。
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKSplitRGBAImage

【機能】
RGBA の 32 ビットイメージから RGB と A のイメージに分割します。

【関数書式】
(1)C++Builder
BOOL IKSplitRGBAImage(HANDLE InHandle, HANDLE *OutHandle1, HANDLE *OutHandle2, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKSplitRGBAImage(InHandle: THandle; var OutHandle1, OutHandle2: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	RGBA の 32 ビットイメージのメモリハンドル
OutHandle1	取得する RGB の 24 ビットイメージのメモリハンドル
OutHandle2	取得する A(アルファチャンネル)の 8 ビットグレースケールイメージのメモリハンドル
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

【参照】
「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」の IKPngFileLoadEx,IKPngFileSaveEx

IKStartDibAccess

【機能】
DIB へのアクセス処理を開始します。

【関数書式】
(1)C++Builder
```
BOOL IKStartDibAccess(HANDLE hDib, PTR_IKDIB_INFO stpDibInfo);
```
(2)Delphi
```
function IKStartDibAccess(hDib: THandle; var stpDibInfo: IKDIB_INFO): LongBool;
```

【引数】

名称	内容
hDib	ラスタイメージのメモリハンドル(1,4,8,16,24,32 ビットイメージ)
stpDibInfo	取得する DIB 情報の構造体(ユーザ定義型)変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
IKEndDibAccess と対で使用します。
IKDIB_INFO については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」の IKDIB_INFO のメンバー変数の説明**をご覧ください。

処理の流れとしては
```
IKStartDibAccess
      |
IKGetDibPixel, IKSetDibPixel
      |
IKEndDibAccess
```
となります。

読み込んだイメージ(1,16,24,32 ビット)をビット反転するコード例:
(1)C++Builder
```
HANDLE ImgHandle;
IKIMAGE_INFO ImageInfo;
IKDIB_INFO DibInfo;
BOOL Ret;
int i, j;

ImgHandle = IKFileLoad("newtone.bmp", 0, 0, 0, 0, NULL, NULL, NULL);
if (ImgHandle == NULL) return;
Ret = IKGetImageType(ImgHandle, &ImageInfo);
Ret = IKStartDibAccess(ImgHandle, &DibInfo);

for (i = 0; i < ImageInfo.Height; i++)
{
   for (j = 0; j < ImageInfo.Width; j++)
   {
      Ret = IKGetDibPixel(&DibInfo, j, i);
      if (ImageInfo.BitCount == 1)
      {
         if (DibInfo.PalNo != 0)
            DibInfo.PalNo = 0;
         else
```

```
                DibInfo.PalNo = 1;
            } else (ImageInfo.BitCount > 8) {
                DibInfo.Blue = 255 - DibInfo.Blue;
                DibInfo.Green = 255 - DibInfo.Green;
                DibInfo.Red = 255 - DibInfo.Red;
            }
            Ret = IKSetDibPixel(&DibInfo, j, i);
        }
    }
    Ret = IKEndDibAccess(ImgHandle, &DibInfo);

    Ret = IKBmpFileSave("newtone.bmp", ImgHandle, FALSE, 0, NULL, NULL, NULL);
    IKFreeMemory(ImgHandle);
```

(2)Delphi

```
    Handle: THandle;
    ImageInfo: IKIMAGE_INFO;
    DibInfo: IKDIB_INFO;
    Ret: LongBool;
    i, j: Integer;

    Handle := IKFileLoad('newtone.bmp', 0, 0, 0, 0, nil, nil, nil);
    if Handle = 0 then Exit;
    Ret := IKGetImageType(Handle, ImageInfo);
    Ret := IKStartDibAccess(Handle, DibInfo);

    for i := 0 to ImageInfo.Height - 1 do
    begin
      for j := 0 to ImageInfo.Width - 1 do
      begin
        Ret := IKGetDibPixel(DibInfo, j, i);
        if ImageInfo.BitCount = 1 then
        begin
          if DibInfo.PalNo <> 0 then
            DibInfo.PalNo := 0
          else
            DibInfo.PalNo := 1;
        end
        else if ImageInfo.BitCount > 8 then
        begin
          DibInfo.Blue := 255 - DibInfo.Blue;
          DibInfo.Green := 255 - DibInfo.Green;
          DibInfo.Red := 255 - DibInfo.Red;
        end;
        Ret := IKSetDibPixel(DibInfo, j, i);
      end;
    end;
    Ret := IKEndDibAccess(Handle, DibInfo);

    Ret := IKBmpFileSave('newtone.bmp', Handle, False, 0, nil, nil, nil);
    IKFreeMemory(Handle);
```

IKUnifyColor

【機能】
色のばらつきを修正します。

【関数書式】
(1)C++Builder
HANDLE IKUnifyColor(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE Red, BYTE Green, BYTE Blue, BYTE Error, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKUnifyColor(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Red, Green, Blue, Error: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Red	修正対象となる赤(0～255)
Green	修正対象となる緑(0～255)
Blue	修正対象となる青(0～255)
Error	各 RGB の誤差範囲(0～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
指定した Red,Green,Blue に対して±Error の範囲の色を Red,Green,Blue に置き換えます。例えば Red,Green,Blue に 128、Error に 1 を設定した場合は、Red,Green,Blue の 127 から 129 の範囲の色をそれぞれを 128 に置き換えます。
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKWaves

【機能】
ラスタイメージにさざ波効果を施します。

【関数書式】
(1)C++Builder
HANDLE IKWaves(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BYTE Direction, BYTE WaveType, int Period, int Amp, double Phase, BYTE Edges, BYTE Red, BYTE Green, BYTE Blue, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKWaves(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Direction, WaveType: Byte; Period, Amp: Integer; Phase: Double; Edges, Red, Green, Blue: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Direction	波の方向(0:横,1:縦)
WaveType	波の種類(0:sin,1:saw)
Period	波の波長(0 以上)
Amp	波の大きさ(0 以上)
Phase	波の位相(0.0～1.0)
Edges	エッジの種類(0:背景色で塗る, 1:隣接したピクセルで埋める, 2:丸める)
Red	背景色の赤(0～255)
Green	背景色の緑(0～255)
Blue	背景色の青(0～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。
イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
Red,Green,Blue は Edges が 0 の場合に有効です。

引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKWhirlPinch

【機能】
ラスタイメージにねじりつまみ効果を施します。

【関数書式】
(1)C++Builder
HANDLE IKWhirlPinch(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, double Whirl, double Pinch, double Radius, BYTE Red, BYTE Green, BYTE Blue, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKWhirlPinch(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Whirl, Pinch, Radius: Double; Red, Green, Blue: Byte; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、8 ビットグレースケール,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Whirl	ねじる角度(度単位)
Pinch	つまみの度合い
Radius	ねじりの大きさ(0 以上)
Red	背景色の赤(0～255)
Green	背景色の緑(0～255)
Blue	背景色の青(0～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKYCCGamma

【機能】
ラスタイメージの YCrCb 値のガンマ補正を行います。

【関数書式】
(1)C++Builder
HANDLE IKYCCGamma(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, double Yb, double Cr, double Cb, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKYCCGamma(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Yb, Cr, Cb: Double; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Yb	輝度値のガンマ係数
Cr	色相 Cr のガンマ係数
Cb	色相 Cb のガンマ係数
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
処理後のピクセル値を y、基のピクセル値を x とすると
(1)Bright ＞ 0 の場合

$$y = (x/255)^{Bright+1} \times 255$$

(2)Bright ＜ 0 の場合

$$y = (x/255)^{-1/Bright-1} \times 255$$

となります。

Bright は Yb、Cr、Cb のいずれかになります。Yb、Cr、Cb が 0 の場合は処理の前後で変化ありません。
Yb、Cr、Cb が＋の場合はレベルが下がり、－の場合はレベルが上がります。

マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、

SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
1,4,8 ビットイメージの場合は、SelectMode が 0～3 のいずれが設定されていてもイメージ全体に対して処理を行います。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKYCCLevel

【機能】
ラスタイメージの YCrCb 値の加減処理を行います。

【関数書式】
(1)C++Builder
HANDLE IKYCCLevel(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, int Yb, int Cr, int Cb, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKYCCLevel(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Yb, Cr, Cb: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Yb	輝度値の加減(-255～255)
Cr	色相 Cr の加減(-255～255)
Cb	色相 Cb の加減(-255～255)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Yb、Cr、Cb のそれぞれの値が大きくなると該当する成分が明るくなり、小さくなると該当する成分が暗くなります。また、Yb、Cr、Cb が 0 の場合は処理の前後で変化ありません。
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
1,4,8 ビットイメージの場合は、SelectMode が 0～3 のいずれが設定されていてもイメージ全体に対して処理を行います。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKYCCRev

【機能】
ラスタイメージの YCrCb 値の反転処理を行います。

【関数書式】
(1)C++Builder
HANDLE IKYCCRev(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, BOOL Yb, BOOL Cr, BOOL Cb, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKYCCRev(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; Yb, Cr, Cb: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
Yb	輝度 Y の反転(False(0): 反転なし, True(0 以外): 反転あり)
Cr	色相 Cr の反転(False(0): 反転なし, True(0 以外): 反転あり)
Cb	色相 Cb の反転(False(0): 反転なし, True(0 以外): 反転あり)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Yb、Cr、Cb を True にするとそれぞれの YCrCb 値を反転します。
例えば、Yb が 255 の場合は 0 となります。(Yb = 255 − Yb)
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
1,4,8 ビットイメージの場合は、SelectMode が 0～3 のいずれが設定されていてもイメージ全体に対して処理を行います。
引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義**をご覧ください。

IKYCCSpline

【機能】
ラスタイメージの YCrCb 値のスプライン補正を行います。

【関数書式】
(1)C++Builder
HANDLE IKYCCSpline(PTR_IKSELECT_IMAGE SrcHandle, BYTE SelectMode, LPPOINT lpPoint, int Points, LPRECT Rect, BOOL InOut, LPPOINT lpSpPoint, int SpPoints, BOOL Yb, BOOL Cr, BOOL Cb, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKYCCSpline(var SrcHandle: IKSELECT_IMAGE; SelectMode: Byte; var lpPoint: TPoint; Points: Integer; var Rect : TRect; InOut: LongBool; var lpSpPoint: TPoint; SpPoints: Integer; Yb, Cr, Cb: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
SrcHandle	ラスタイメージとマスクハンドルの構造体 **hImgBmh** には、1,4,8,16,24,32 ビットイメージのいずれかを設定すること
SelectMode	選択範囲の種類の指定(0:マスクハンドル,1:イメージ全体, 2:多角形, 3:円形)
lpPoint	多角形の座標を指定する構造体変数の配列(SelectMode が 2 の時有効) (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(SelectMode が 2 の時有効)
Rect	円に外接する四角形の座標を指定する構造体変数(SelectMode が 3 の時有効)
InOut	選択範囲の内外の設定(多角形もしくは円形のみ有効) False(0):選択範囲の外側(境界含まない)、True(0 以外):選択範囲の内側(境界含む)
lpSpPoint	スプラインの通過座標を指定する構造体変数の配列(値の設定方法は、lpPoint と同じ)
SpPoints	lpSpPoint の配列の数(3～10 が有効)
Yb	輝度 Y の成分に対して処理を行うかどうかの設定(False(0):しない　True(0 以外):する)
Cr	色相 Cr の成分に対して処理を行うかどうかの設定(False(0):しない　True(0 以外):する)
Cb	色相 Cb の成分に対して処理を行うかどうかの設定(False(0):しない　True(0 以外):する)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
処理後のラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
入力されたイメージデータの YCrCb 値に対してスプライン曲線により変化させることができます。
使い方は Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll の **IKRGBSpline** の「使用例」を参照してください。
マスクハンドルを基に処理を行いたい場合は、SelectMode に 0 を設定し、マスクハンドルに有効な値を設定します。イメージ全体に対して処理を行いたい場合は、SelectMode に 1 を設定します。選択した多角形に対して処理を行いたい場合には、SelectMode に 2 を設定し、lpPoint と Points にそれぞれ有効な値を設定します。その場合 Points に 2 以上を設定する必要があります。選択した円形に対して処理を行いたい場合には、SelectMode に 3 を設定し、Rect に有効な値を設定します。
lpPoint と Rect のメンバーに座標を設定する場合は、ピクセル単位で設定してください。
SelectMode が 2 以外の場合には、lpPoint にダミーの配列を、Points には 0 を与えてください。SelectMode が 3 以外の場合には、Rect にダミーの変数を与えてください。
1,4,8 ビットイメージの場合は、SelectMode が 0～3 のいずれが設定されていてもイメージ全体に対して処理を行います。
lpSpPoint のメンバーには 0～255 の間の値を設定してください。

引数として与えた IKSELECT_IMAGE のメンバーのメモリハンドルは解放されずにそのまま残ります。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」のユーザ関数の定義をご覧ください。

1－3. Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll

イメージファイルの入出力機能を提供します。

●DLL 関数コマンド一覧（アルファベット順）

関数名	内容
IKBmpFileLoad	BMP 形式のファイルからラスタイメージをメモリ上にロード
IKBmpFileLoadMem	BMP 形式の Raw データからラスタイメージをメモリ上にロード
IKBmpFileSave	ラスタイメージを BMP 形式でファイルへ保存
IKBmpFileSaveMem	ラスタイメージを BMP 形式で Raw データへ保存
IKCMYKBmpPlaneFileLoad	CMYK プレーン毎に保存してある BMP ファイルをメモリ上にロード
IKCMYKBmpPlaneFileSave	ラスタイメージを CMYK プレーン毎に BMP 形式でファイルへ保存
IKDxfFileLoad	DXF 形式のファイルからベクトルイメージをメモリ上にロード
IKDxfFileLoadMem	DXF 形式の Raw データからベクトルイメージをメモリ上にロード
IKDxfFileSave	ベクトルイメージを DXF 形式でファイルへ保存
IKDxfFileSaveMem	ベクトルイメージを DXF 形式で Raw データへ保存
IKEmfFileLoad	EMF 形式のファイルからベクトルイメージをメモリ上にロード
IKEmfFileLoadMem	EMF 形式の Raw データからベクトルイメージをメモリ上にロード
IKEmfFileSave	ベクトルイメージを EMF 形式でファイルへ保存
IKEmfFileSaveMem	ベクトルイメージを EMF 形式で Raw データへ保存
IKFileLoad	イメージファイルからイメージをメモリ上にロード
IKFileLoadAsRawData	イメージファイルから Raw データをメモリ上にロード
IKFileLoadMem	Raw データからラスタイメージもしくはベクトルイメージをメモリ上にロード
IKFileSaveAsRawData	Raw データをファイルへ保存
IKFileType	イメージファイルの情報を取得
IKFileTypeMem	Raw データからイメージの情報を取得
IKFpxFileLoad	FPX 形式のファイルからラスタイメージをメモリ上にロード
IKFpxFileLoadMem	FPX 形式の Raw データからラスタイメージをメモリ上にロード
IKFpxFileSave	ラスタイメージを FPX 形式でファイルへ保存
IKFpxFileSaveMem	ラスタイメージを FPX 形式で Raw データへ保存
IKFTPConnect	FTP サーバへの接続
IKFTPDeleteFile	FTP サーバに存在するファイルを削除（接続と切断も行う）
IKFTPDeleteFileEx	FTP サーバに存在するファイルを削除
IKFTPDisconnect	接続している FTP サーバの切断
IKFTPGetFile	FTP サーバからファイルを取得（接続と切断も行う）
IKFTPGetFileEx	FTP サーバからファイルを取得
IKFTPPutFile	FTP サーバへファイルを転送（接続と切断も行う）
IKFTPPutFileEx	FTP サーバへファイルを転送
IKFTPRenameFile	FTP サーバに存在するファイルの名称を変更（接続と切断も行う）
IKFTPRenameFileEx	FTP サーバに存在するファイルの名称を変更
IKGifFileLoad	GIF 形式のファイルからラスタイメージをメモリ上にロード
IKGifFileLoadMem	GIF 形式の Raw データからラスタイメージをメモリ上にロード
IKGifFileSave	ラスタイメージを GIF 形式でファイルへ保存
IKGifFileSaveMem	ラスタイメージを GIF 形式で Raw データへ保存
IKHTTPConnect	HTTP(S)サーバへの接続
IKHTTPDisconnect	接続している HTTP(S)サーバの切断
IKHTTPGetFile	HTTP(S)サーバからファイルを取得（接続と切断も行う）
IKHTTPGetFileEx	HTTP(S)サーバからファイルを取得
IKHTTPPutFile	HTTP(S)サーバへファイルを転送（接続と切断も行う）
IKHTTPPutFileEx	HTTP(S)サーバへファイルを転送
IKJ2kFileLoad	JPEG2000 形式のファイルからラスタイメージをメモリ上にロード
IKJ2kFileLoadMem	JPEG2000 形式の Raw データからラスタイメージをメモリ上にロード
IKJ2kFileSave	ラスタイメージを JPEG2000 形式でファイルへ保存
IKJ2kFileSaveMem	ラスタイメージを JPEG2000 形式で Raw データへ保存
IKJpegExifInfo	Exif 形式のファイルから画像の情報を取得
IKJpegExifInfoMem	Exif 形式のメモリハンドルから画像の情報を取得

IKJpegFileLoad	JPEG 形式のファイルからラスタイメージをメモリ上にロード
IKJpegFileLoadMem	JPEG 形式の Raw データからラスタイメージをメモリ上にロード
IKJpegFileSave(Ex)	ラスタイメージを JPEG 形式でファイルへ保存(Ex は Exif 情報も保存)
IKJpegFileSave(Ex)Mem	ラスタイメージを JPEG 形式で Raw データへ保存(Ex は Exif 情報も保存)
IKOpenFileDialog	プレビュー付きのファイル選択ダイアログ(オープン)
IKOpenFileDlg	プレビュー付きのファイル選択ダイアログ(オープン)
IKPcxFileLoad	PCX 形式のファイルからラスタイメージをメモリ上にロード
IKPcxFileLoadMem	PCX 形式の Raw データからラスタイメージをメモリ上にロード
IKPcxFileSave	ラスタイメージを PCX 形式でファイルへ保存
IKPcxFileSaveMem	ラスタイメージを PCX 形式で Raw データへ保存
IKPDFAddImage	PDF 作成のイメージの追加処理
IKPDFAddPage	PDF 作成のページの追加処理
IKPDFEnd	PDF 作成の終了処理
IKPDFStart	PDF 作成の開始処理
IKPngFileLoad(Ex)	PNG 形式のファイルからラスタイメージをメモリ上にロード
IKPngFileLoad(Ex)Mem	PNG 形式の Raw データからラスタイメージをメモリ上にロード
IKPngFileSave(Ex)	ラスタイメージを PNG 形式でファイルへ保存
IKPngFileSave(Ex)Mem	ラスタイメージを PNG 形式で Raw データへ保存
IKRGBBmpPlaneFileLoad	RGB プレーン毎に保存してある BMP ファイルをメモリ上にロード
IKRGBBmpPlaneFileSave	ラスタイメージを RGB プレーン毎に BMP 形式でファイルへ保存
IKSaveFileDialog	プレビュー付きのファイル選択ダイアログ(保存)
IKSaveFileDlg(Ex)	プレビュー付きのファイル選択ダイアログ(保存)
IKSvgFileLoad	SVG 形式のファイルからベクトルイメージをメモリ上にロード
IKSvgFileLoadMem	SVG 形式の Raw データからベクトルイメージをメモリ上にロード
IKSvgFileSave	ベクトルイメージを SVG 形式でファイルへ保存
IKSvgFileSaveMem	ベクトルイメージを SVG 形式で Raw データへ保存
IKSxfP21FileLoad	SXF(p21)形式のファイルからベクトルイメージをメモリ上にロード
IKSxfP21FileLoadMem	SXF(p21)形式の Raw データからベクトルイメージをメモリ上にロード
IKSxfP21FileSave	ベクトルイメージを SXF(p21)形式でファイルへ保存
IKSxfP21FileSaveMem	ベクトルイメージを SXF(p21)形式で Raw データへ保存
IKSxfSfcFileLoad	SXF(Sfc)形式のファイルからベクトルイメージをメモリ上にロード
IKSxfSfcFileLoadMem	SXF(Sfc)形式の Raw データからベクトルイメージをメモリ上にロード
IKSxfSfcFileSave	ベクトルイメージを SXF(Sfc)形式でファイルへ保存
IKSxfSfcFileSaveMem	ベクトルイメージを SXF(Sfc)形式で Raw データへ保存
IKTiffFileLoad	TIFF 形式のファイルからラスタイメージをメモリ上にロード
IKTiffFileLoadMem	TIFF 形式の Raw データからラスタイメージをメモリ上にロード
IKTiffFileSave	ラスタイメージを TIFF 形式でファイルへ保存
IKTiffFileSaveMem	ラスタイメージを TIFF 形式で Raw データへ保存
IKWmfFileLoad	WMF 形式のファイルからベクトルイメージをメモリ上にロード
IKWmfFileLoadMem	WMF 形式の Raw データからベクトルイメージをメモリ上にロード
IKWmfFileSave	ベクトルイメージを WMF 形式でファイルへ保存
IKWmfFileSaveMem	ベクトルイメージを WMF 形式で Raw データへ保存
IKYCCBmpPlaneFileLoad	YCrCb プレーン毎に保存してある BMP ファイルをメモリ上にロード
IKYCCBmpPlaneFileSave	ラスタイメージを YCrCb プレーン毎に BMP 形式でファイルへ保存

●構造体(ユーザ定義型)の定義

IKFILE_INFO： ファイル情報を表します(IKFileType,IKFileTypeMem で使用します)。

(1)C++Builder

```
typedef struct {
    short       BitCount;
    long        Xdpi;
    long        Ydpi;
    long        Width;
    long        Height;
    long        WidthByte;
    long        ImageSize;
    short       FileType;
    short       Interlace;
```

```
    short       MaxPage;
    long        FileSize;
    char        Comment[2048];
    WORD        CommentLen;
    short       CreationTimeYear;
    short       CreationTimeMonth;
    short       CreationTimeDay;
    short       CreationTimeHour;
    short       CreationTimeMinute;
    short       CreationTimeSecond;
    short       LastAccessTimeYear;
    short       LastAccessTimeMonth;
    short       LastAccessTimeDay;
    short       LastAccessTimeHour;
    short       LastAccessTimeMinute;
    short       LastAccessTimeSecond;
    short       LastWriteTimeYear;
    short       LastWriteTimeMonth;
    short       LastWriteTimeDay;
    short       LastWriteTimeHour;
    short       LastWriteTimeMinute;
    short       LastWriteTimeSecond;
  } IKFILE_INFO;
  typedef IKFILE_INFO * PTR_IKFILE_INFO;
```

(2)Delphi

```
  type
    IKFILE_INFO = Record
      BitCount:              Smallint;
      Xdpi:                  Longint;
      Ydpi:                  Longint;
      Width:                 Longint;
      Height:                Longint;
      WidthByte:             Longint;
      ImageSize:             Longint;
      FileType:              Smallint;
      Interlace:             Smallint;
      MaxPage:               Smallint;
      FileSize:              Longint;
      Comment:               array [0..2047] of AnsiChar;
      CommentLen:            Word;
      CreationTimeYear:      Smallint;
      CreationTimeMonth:     Smallint;
      CreationTimeDay:       Smallint;
      CreationTimeHour:      Smallint;
      CreationTimeMinute:    Smallint;
      CreationTimeSecond:    Smallint;
      LastAccessTimeYear:    Smallint;
      LastAccessTimeMonth:   Smallint;
      LastAccessTimeDay:     Smallint;
      LastAccessTimeHour:    Smallint;
      LastAccessTimeMinute:  Smallint;
      LastAccessTimeSecond:  Smallint;
      LastWriteTimeYear:     Smallint;
      LastWriteTimeMonth:    Smallint;
      LastWriteTimeDay:      Smallint;
      LastWriteTimeHour:     Smallint;
      LastWriteTimeMinute:   Smallint;
      LastWriteTimeSecond:   Smallint;
```

```
    end;
```

IKFileType 関数を実行するとそれぞれのメンバーに値が設定されます。MaxPage、FileSize 以外のメンバーは IKFileType 関数で指定されたページの情報となります。ただし、TIFF 形式を除く Comment はファイル単位の情報となります。

BitCount: イメージの 1 ピクセルあたりのビット数を表します。 *1
1:2 値、4:16 色、8:256 色、16:16 ﾋﾞｯﾄｶﾗｰ、24:24 ﾋﾞｯﾄｶﾗｰ、32:32 ﾋﾞｯﾄｶﾗｰ

Xdpi: イメージの横方向(X 方向)の 1 インチあたりのピクセル数を表します。
Ydpi: イメージの縦方向(Y 方向)の 1 インチあたりのピクセル数を表します。
Width: イメージの幅のピクセル数を表します。
Height: イメージの縦のピクセル数を表します。
WidthByte: イメージの 1 ラインのバイト数を表します。 *1
ImageSize: イメージサイズのバイト数を表します。 *1
FileType: ファイルのタイプを表します。

1:BMP/DIB	2:BMP/DIB(RLE4)	3:BMP/DIB(RLE8)	4:JPEG(基本 DCT)
5:JPEG(ﾌﾟﾛｸﾞﾚｯｼﾌﾞ DCT)	6:GIF(87a)	7:GIF(89a)	8:TIFF(非圧縮)
9:TIFF(CCITTRLE)	10:TIFF(GROUP3-1D)	11:TIFF(GROUP3-2D)	12:TIFF(GROUP4)
13:TIFF(PACKBITS)	14:TIFF(LZW)	15:PNG	16:FPX(非圧縮)
17:FPX(Single Color)	18:FPX(JPEG)	19:PCX	20:WMF
21:EMF	22:DXF	23:Exif(JPEG)	24:SVG
25:JPEG2000	26:JPEG2000(Code Stream)	27:SXF(p21)	28:SXF(sfc)
29:TIFF(JPEG)			

TIFF の GROUP3-1D は MH、GROUP3-2D は MR、GROUP4 は MMR と同じ形式です。
Exif はバージョン 2.3 の一部のタグにも対応しています。0 の場合は未対応の形式です。

Interlace: インタレース(0:なし 1:あり)を表します。GIF と PNG が対象。 *1
MaxPage: イメージのページ数を表します。
FileSize: イメージファイルのサイズを表します。
Comment: イメージのコメントを表します(コメントの形式については、ファイル保存コマンドをご覧ください)。 *1 *3
CommentLen: Comment に含まれる文字列の長さを表します。 *2

CreationTimeYear: ファイルの作成日時 - 年を示します。 *2
CreationTimeMonth: ファイルの作成日時 - 月を示します。 *2
CreationTimeDay: ファイルの作成日時 - 日を示します。 *2
CreationTimeHour: ファイルの作成日時 - 時を示します。 *2
CreationTimeMinute: ファイルの作成日時 - 分を示します。 *2
CreationTimeSecond: ファイルの作成日時 - 秒を示します。 *2
LastAccessTimeYear: ファイルのアクセス日時 - 年を示します。 *2
LastAccessTimeMonth: ファイルのアクセス日時 - 月を示します。 *2
LastAccessTimeDay: ファイルのアクセス日時 - 日を示します。 *2
LastAccessTimeHour: ファイルのアクセス日時 - 時を示します。 *2
LastAccessTimeMinute: ファイルのアクセス日時 - 分を示します。 *2
LastAccessTimeSecond: ファイルのアクセス日時 - 秒を示します。 *2
LastWriteTimeYear: ファイルの更新日時 - 年を示します。 *2
LastWriteTimeMonth: ファイルの更新日時 - 月を示します。 *2
LastWriteTimeDay: ファイルの更新日時 - 日を示します。 *2
LastWriteTimeHour: ファイルの更新日時 - 時を示します。 *2
LastWriteTimeMinute: ファイルの更新日時 - 分を示します。 *2
LastWriteTimeSecond: ファイルの更新日時 - 秒を示します。 *2

*1 ベクトルイメージでは使用しません。
*2 ImageKit7 で追加されたメンバー変数です。ImageKit5/6 から移行する場合は注意してください。
*3 ImageKit7 でサイズが変更されました。ImageKit5/6 から移行する場合は注意してください。

IMAGE_TAG: Exif ファイルの主画像、サムネイル画像の情報を表します。

(1)C++Builder

```
  typedef struct {
```

```
    unsigned short      Compression;
    unsigned short      Orientation;
    unsigned short      YCbCrPositioning;
    unsigned long       XResolution[2];
    unsigned long       YResolution[2];
    unsigned short      ResolutionUnit;
    char                DateTime[20];
    char                ImageDescription[257];
    WORD                ImageDescriptionLen;
    char                Make[257];
    WORD                MakeLen;
    char                Model[257];
    WORD                ModelLen;
    char                Software[257];
    WORD                SoftwareLen;
    char                Artist[257];
    WORD                ArtistLen;
    char                Copyright[257];
    WORD                CopyrightLen;
    HANDLE              ThumbImageHandle;
  } IMAGE_TAG;
  typedef IMAGE_TAG * PTR_IMAGE_TAG;
```

(2)Delphi

```
  type
    IMAGE_TAG = Record
      Compression:          Word;
      Orientation:          Word;
      YCbCrPositioning:     Word;
      XResolution:          array[0..1] of DWORD;
      YResolution:          array[0..1] of DWORD;
      ResolutionUnit:       Word;
      DateTime:             array[0..19] of AnsiChar;
      ImageDescription:     array[0..256] of AnsiChar;
      ImageDescriptionLen:  Word;
      Make:                 array[0..256] of AnsiChar;
      MakeLen:              Word;
      Model:                array[0..256] of AnsiChar;
      ModelLen:             Word;
      Software:             array[0..256] of AnsiChar;
      SoftwareLen:          Word;
      Artist:               array[0..256] of AnsiChar;
      ArtistLen:            Word;
      Copyright:            array[0..256] of AnsiChar;
      CopyrightLen:         Word;
      ThumbImageHandle:     THandle;
    end;
```

Compression: 画像の圧縮の種類を示します。(1:非圧縮、6:JPEG 圧縮 ※サムネイル画像のみ)
Orientation: 行と列の観点から見た画像の方向を示します。
YCbCrPositioning: 輝度サンプルに対するクロマサンプルの相対的配置を特定します。(1:中心、2:一致)

XResolution: イメージの横方向の 1**ResolutionUnit** あたりの画素数を示します。
横方向の解像度(単位は **ResolutionUnit** 値) = **XResolution[0]** / **XResolution[1]**

YResolution: イメージの縦方向の 1**ResolutionUnit** あたりの画素数を示します。
横方向の解像度(単位は **ResolutionUnit** 値) = **YResolution[0]** / **YResolution[1]**

ResolutionUnit: 解像度の単位を示します。(2:インチ、3:センチメートル)

DateTime:	ファイル作成日時を示します。(フォーマットは"YYYY:MM:DD HH:MM:SS")
ImageDescription:	画像タイトルを示します。 *2
ImageDescriptionLen:	**ImageDescription** に含まれる文字列の長さを表します。 *1
Make:	画像入力機器のメーカ名を示します。 *2
MakeLen:	**Make** に含まれる文字列の長さを表します。 *1
Model:	画像入力機器のモデル名を示します。 *2
ModelLen:	**Model** に含まれる文字列の長さを表します。 *1
Software:	使用ソフトウェア名を示します。 *2
SoftwareLen:	**Software** に含まれる文字列の長さを表します。 *1
Artist:	作者名を示します。 *2
ArtistLen:	**Artist** に含まれる文字列の長さを表します。 *1
Copyright:	撮影著作権者/編集著作権者を示します。 *2
CopyrightLen:	**Copyright** に含まれる文字列の長さを表します。 *1
ThumbImageHandle:	サムネイル画像のメモリハンドルを示します。(※サムネイル画像のみ)

*1 ImageKit7 で追加されたメンバー変数です。ImageKit6 から移行する場合は注意してください。
*2 ImageKit8 でサイズが変更されました。ImageKit6/7 から移行する場合は注意してください。

EXIF_TAG: Exif のタグ情報を表します。

(1)C++Builder

```
typedef struct {
    char                Exif_Version[5];
    char                FlashPix_Version[5];
    unsigned short      ColorSpace;
    unsigned long       PixelXDimension;
    unsigned long       PixelYDimension;
    char                ComponentsConfiguration[5];
    unsigned long       CompressedBitsPerPixel[2];
    char                MakerNote[257];
    WORD                MakerNoteLen;
    char                UserComment[257];
    WORD                UserCommentLen;
    char                UserCommentID[8];
    char                DateTimeOriginal[20];
    char                DateTimeDigitized[20];
    char                SubSecTime[129];
    WORD                SubSecTimeLen;
    char                SubSecTimeOriginal[129];
    WORD                SubSecTimeOriginalLen;
    char                SubSecTimeDigitized[129];
    WORD                SubSecTimeDigitizedLen;
    unsigned long       ExposureTime[2];
    long                ShutterSpeedValue[2];
    unsigned long       ApertureValue[2];
    long                BrightnessValue[2];
    long                ExposureBiasValue[2];
    unsigned long       MaxApertureValue[2];
    unsigned long       SubjectDistance[2];
    unsigned short      MeteringMode;
    unsigned short      LightSource;
    unsigned short      Flash;
    unsigned long       FocalLength[2];
    unsigned long       FNumber[2];
    unsigned short      ExposureProgram;
    char                SpectralSensitivity[257];
    WORD                SpectralSensitivityLen;
    unsigned short      ISOSpeedRatings;
    unsigned long       FlashEnergy[2];
```

```
    unsigned long           FocalPlaneXResolution[2];
    unsigned long           FocalPlaneYResolution[2];
    unsigned short          FocalPlaneResolutionUnit;
    unsigned short          SubjectLocationX;
    unsigned short          SubjectLocationY;
    unsigned long           ExposureIndex[2];
    unsigned short          SensingMethod;
    BYTE                    FileSource;
    BYTE                    SceneType;
    unsigned short          WhiteBalance;
    unsigned short          SceneCaptureType;
    unsigned short          Contrast;
    unsigned short          Saturation;
    unsigned short          Sharpness;
  } EXIF_TAG;
  typedef EXIF_TAG * PTR_EXIF_TAG;
```

(2)Delphi

```
  type
    EXIF_TAG = Record
      Exif_Version:             array[0..4] of AnsiChar;
      FlashPix_Version:         array[0..4] of AnsiChar;
      ColorSpace:               Word;
      PixelXDimension:          DWORD;
      PixelYDimension:          DWORD;
      ComponentsConfiguration:  array[0..4] of AnsiChar;
      CompressedBitsPerPixel:   array[0..1] of DWORD;
      MakerNote:                array[0..256] of AnsiChar;
      MakerNoteLen:             Word;
      UserComment:              array[0..256] of AnsiChar;
      UserCommentLen:           Word;
      UserCommentID:            array[0..7] of AnsiChar;
      DateTimeOriginal:         array[0..19] of AnsiChar;
      DateTimeDigitized:        array[0..19] of AnsiChar;
      SubSecTime:               array[0..128] of AnsiChar;
      SubSecTimeLen:            Word;
      SubSecTimeOriginal:       array[0..128] of AnsiChar;
      SubSecTimeOriginalLen:    Word;
      SubSecTimeDigitized:      array[0..128] of AnsiChar;
      SubSecTimeDigitizedLen:   Word;
      ExposureTime:             array[0..1] of DWORD;
      ShutterSpeedValue:        array[0..1] of Longint;
      ApertureValue:            array[0..1] of DWORD;
      BrightnessValue:          array[0..1] of Longint;
      ExposureBiasValue:        array[0..1] of Longint;
      MaxApertureValue:         array[0..1] of DWORD;
      SubjectDistance:          array[0..1] of DWORD;
      MeteringMode:             Word;
      LightSource:              Word;
      Flash:                    Word;
      FocalLength:              array[0..1] of DWORD;
      FNumber:                  array[0..1] of DWORD;
      ExposureProgram:          Word;
      SpectralSensitivity:      array[0..256] of AnsiChar;
      SpectralSensitivityLen:   Word;
      ISOSpeedRatings:          Word;
      FlashEnergy:              array[0..1] of DWORD;
      FocalPlaneXResolution:    array[0..1] of DWORD;
      FocalPlaneYResolution:    array[0..1] of DWORD;
```

```
    FocalPlaneResolutionUnit:    Word;
    SubjectLocationX:            Word;
    SubjectLocationY:            Word;
    ExposureIndex:               array[0..1] of DWORD;
    SensingMethod:               Word;
    FileSource:                  Byte;
    SceneType:                   Byte;
    WhiteBalance:                Word;
    SceneCaptureType:            Word;
    Contrast:                    Word;
    Saturation:                  Word;
    Sharpness:                   Word;
  end;
```

バージョンに関するタグ

Exif_Version: Exif の対応バージョンを示します。(例:"0210")

FlashPix_Version: FPXR ファイルの FlashPix フォーマットへの対応バージョンを示します。(例:"0100")

画像データの特性に関するタグ

ColorSpace: 色空間を示す情報です。(1:sRGB)

画像データの構成に関するタグ

PixelXDimension: 実効画像幅を示します。

PixelYDimension: 実効画像高さを示します。

ComponentsConfiguration: 各コンポーネントのチャネルを第 1 コンポーネントから第 4 コンポーネントの順に示します。

CompressedBitsPerPixel: 画像圧縮時に設定された圧縮モードを示します。

圧縮モード(単位は bit/pixel) = **CompressedBitsPerPixel[0]** / **CompressedBitsPerPixel[1]**

ユーザ情報に関するタグ

MakerNote: Exif ライターのメーカが個別の情報を記入するタグです。 *2

MakerNoteLen: **MakerNote** に含まれる文字列の長さを表します。 *1

UserComment: 画像に対して Exif ユーザがキーワードやコメントを書き込むためのタグです。 *2

UserCommentLen: **UserComment** に含まれる文字列の長さを表します。 *1

UserCommentID: **UserComment** に書かれる文字コードを判別するために、識別コードを 8 バイト固定で記入し、余った領域には NULL(0x00)でパディングします。

文字コード	コード記入方法(8 バイト)	リファレンス
ASCII	0x41,0x53,0x43,0x49,0x49,0x00,0x00,0x00	ITU-T T.50 IA5
JIS	0x4A,0x49,0x53,0x00,0x00,0x00,0x00,0x00	JIS X0208-1990
Unicode	0x55,0x4E,0x49,0x43,0x4F,0x44,0x45,0x00	Unicode Standard
Undefined	0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00	Undefined

※16 進表記のため、Delphi は 0x を$に置き換えてください。

日時に関するタグ

DateTimeOriginal: 原画像データの生成された日付と時間(デジタルカメラでは撮影された日付と時間)を示します。(フォーマットは"YYYY:MM:DD HH:MM:SS")

DateTimeDigitized: 画像がデジタルデータ化された日付と時間を示します。(フォーマットは"YYYY:MM:DD HH:MM:SS")

SubSecTime: **IMAGE_TAG** の **DateTime** に関連して時刻を小数点以下の秒単位まで示します。 *2

SubSecTimeLen: **SubSecTime** に含まれる文字列の長さを表します。 *3

SubSecTimeOriginal: **DateTimeOriginal** に関連して時刻を小数点以下の秒単位まで示します。 *2

SubSecTimeOriginalLen: **SubSecTimeOriginalLen** に含まれる文字列の長さを表します。 *3

SubSecTimeDigitized: **DateTimeDigitized** に関連して時刻を小数点以下の秒単位まで示します。 *2

SubSecTimeDigitizedLen: **SubSecTimeDigitizedLen** に含まれる文字列の長さを表します。 *3

撮影条件に関するタグ

ExposureTime: 露出時間を示します。露出時間(単位は秒) = **ExposureTime[0]** / **ExposureTime[1]**

ShutterSpeedValue: シャッタースピードを示します。

シャッタースピード(単位は APEX 値) = **ShutterSpeedValue[0]** / **ShutterSpeedValue[1]**

ApertureValue: レンズの絞り値を示します。

レンズの絞り値（単位は APEX 値） = ApertureValue[0] / ApertureValue[1]

BrightnessValue: 輝度値を示します。一般的な記載範囲は-99.99 から 99.99 です。

輝度値（単位は APEX 値） = BrightnessValue[0] / BrightnessValue[1]

ExposureBiasValue: 露光補正値を示します。一般的な記載範囲は-99.99 から 99.99 です。

露光補正値（単位は APEX 値） = ExposureBiasValue[0] / ExposureBiasValue[1]

MaxApertureValue: レンズの最小 F 値を示します。一般的な記載範囲は 00.00 から 99.99 です。

レンズの最小 F 値（単位は APEX 値） = MaxApertureValue[0] / MaxApertureValue[1]

SubjectDistance: 被写体距離を示します。

被写体距離（単位は m） = SubjectDistance[0] / SubjectDistance [1]

MeteringMode: 測光方式を示します。

（0：不明、1:平均、2：中央重点、3：スポット、4：マルチスポット、5：分割測光、6：部分測光、255：その他）

LightSource: 光源の種類を示します。

（0：不明、1:昼光、2：蛍光灯、3：タングステン（白熱灯）、4:フラッシュ、9：晴天、10：曇天、11:日陰、12:昼光色蛍光灯（D:5700 － 7100K）、13:昼白色蛍光灯（N:4600 － 5500K）、14:白色蛍光灯（W:3800 － 4500K）、15:温白色蛍光灯（WW:3250 － 3800K）、16:電球色蛍光灯（L:2600 － 3250K）、17：標準光 A、18:標準光 B、19：標準光 C、20:D55、21:D65、22:D75、23:D50、24:ISO studio tungsten、255:その他）

Flash: ストロボを使用して画像が取り込まれた時に記録される値です。

ビット 0 はストロボの状態、ビット 1 および 2 はストロボのリターン状態、ビット 3 および 4 はカメラのストロボモード、ビット 5 はストロボ機能の有無、ビット 6 は赤目モードを表わします。

ストロボ発光状態のビットの値（bit0）

0b = ストロボ発光せず、1b = ストロボ発光

ストロボのリターン状態の値（bit1,2）

00b = ストロボのリターン検出機能なし、01b = 予約、10b = ストロボのリターン検出されず

11b = ストロボのリターン検出

カメラのストロボモードの値（bit3,4）

00b = モード不明、01b = 強制発行モード、10b = 強制非発行モード、11b = 自動発行モード

ストロボ機能の有無（bit5）

0b = ストロボ機能有り、1b = ストロボ機能無し

カメラの赤目モードの値（bit6）

0b = 赤目軽減無しまたは不明、1b = 赤目軽減有り

「16 進表記」

0x0000 = ストロボ発光せず、0x0001 = ストロボ発光、0x0005 = ストロボ発光、リターン検出されず

0x0007 = ストロボ発光、リターン検出

※Delphi は 0x を$に置き換えてください。

FocalLength: 撮影レンズの実焦点距離を示します。

撮影レンズの実焦点距離（単位は mm） = FocalLength[0] / FocalLength[1]

FNumber: F ナンバーを示します。

F ナンバー = FNumber[0] / FNumber[1]

ExposureProgram: 撮影時にカメラが使用した露出プログラムのクラスを示します。

（0:未定義、1:マニュアル、2:ノーマルプログラム、3:絞り優先、4:シャッター優先、5：creative プログラム（被写界深度方向にバイアス）、6:action プログラム（シャッタースピード高速側にバイアス）、7:ポートレイトモード（クローズアップ撮影、背景はフォーカス外す）、8:ランドスケープモード（landscape 撮影、背景はフォーカス合う））

SpectralSensitivity: 撮影に用いたカメラの各チャネルのスペクトル感度を示します。

SpectralSensitivityLen: SpectralSensitivity に含まれる文字列の長さを表します。 *1

ISOSpeedRatings: ISO 12232xiv で規定されるカメラまたは入力機器の ISO Speed および ISO Latitude です。

FlashEnergy: 画像が取り込まれた時に使用されたストロボのエネルギーを示します。

ストロボのエネルギー（測定単位は BCPS） = FlashEnergy[0] / FlashEnergy[1]

FocalPlaneXResolution: 焦点面の幅の解像度を示します。

解像度（単位は FocalPlaneResolutionUnit 値） = FocalPlaneXResolution[0] / FocalPlaneXResolution[1]

FocalPlaneYResolution: 焦点面の高さの解像度を示します。

解像度（単位は FocalPlaneResolutionUnit 値） = FocalPlaneYResolution[0] / FocalPlaneYResolution[1]

FocalPlaneResolutionUnit: 焦点面の解像度単位を示します。（2：インチ、3：センチメートル）

SubjectLocationX: 主要被写体のおおよその X 座標値を示します。

SubjectLocationY: 主要被写体のおおよその Y 座標値を示します。

ExposureIndex: 画像が取り込まれた時、カメラまたは入力機器が選択した露出のインデックスを示します。

露出インデックス = ExposureIndex[0] / ExposureIndex[1]

SensingMethod: カメラまたは入力機器で使用される画像センサのタイプを示します。

（1:未定義、2：単板カラーセンサ、3：2 板カラーセンサ、4：3 板カラーセンサ、5:色順次カラーセンサ、7：3 線リニアセンサ、

8:色順次リニアセンサ)

FileSource:	画像のソースを示します。(デジタルカメラで記録する場合には常に 3)
SceneType:	画像のシーンのタイプを示します。(デジタルカメラで記録する場合には常に 1)
WhiteBalance:	撮影時に設定されたホワイトバランスモードを示します *3
SceneCaptureType:	撮影時の被写体種別を示します。 *3
Contrast:	撮影時にカメラが画像に施したコントラスト処理傾向を示します。 *3
Saturation:	撮影時にカメラが画像に施した彩度処理傾向を示します。 *3
Sharpness:	撮影時にカメラが画像に施したシャープネス処理傾向を示します。 *3

APEX は Additive System of Photographic Exposure の略
BCPS は Beam Candle Power Seconds の略

*1 ImageKit7 で追加されたメンバー変数です。ImageKit6 から移行する場合は注意してください。
*2 ImageKit8 でサイズが変更されました。ImageKit6/7 から移行する場合は注意してください。
*3 ImageKit8 で追加されたメンバー変数です。ImageKit6/7 から移行する場合は注意してください。

GPS_TAG: GPS のタグ情報を表します。

(1)C++Builder

```
typedef struct {
    char                GPS_Version[5];
    char                MapDatum[129];
    WORD                MapDatumLen;
    char                Latitude[2];
    unsigned long       LatitudeD[2];
    unsigned long       LatitudeM[2];
    unsigned long       LatitudeS[2];
    char                Longitude[2];
    unsigned long       LongitudeD[2];
    unsigned long       LongitudeM[2];
    unsigned long       LongitudeS[2];
    BYTE                AltitudeRef;
    unsigned long       Altitude[2];
    unsigned long       TimeStampH[2];
    unsigned long       TimeStampM[2];
    unsigned long       TimeStampS[2];
    char                ProcessingMethod[257];
    WORD                ProcessingMethodLen;
    char                ProcessingMethodID[8];
    char                DateStamp[11];
} GPS_TAG;
typedef GPS_TAG * PTR_GPS_TAG;
```

(2)Delphi

```
type
  GPS_TAG = Record
    GPS_Version:        array[0..4] of AnsiChar;
    MapDatum:           array[0..128] of AnsiChar;
    MapDatumLen:        Word;
    Latitude:           array[0..1] of AnsiChar;
    LatitudeD:          array[0..1] of DWORD;
    LatitudeM:          array[0..1] of DWORD;
    LatitudeS:          array[0..1] of DWORD;
    Longitude:          array[0..1] of AnsiChar;
    LongitudeD:         array[0..1] of DWORD;
    LongitudeM:         array[0..1] of DWORD;
    LongitudeS:         array[0..1] of DWORD;
    AltitudeRef:        Byte;
    Altitude:           array[0..1] of DWORD;
```

```
    TimeStampH:          array[0..1] of DWORD;
    TimeStampM:          array[0..1] of DWORD;
    TimeStampS:          array[0..1] of DWORD;
    ProcessingMethod:    array[0..256] of AnsiChar;
    ProcessingMethodLen:Word;
    ProcessingMethodID: array[0..7] of AnsiChar;
    DateStamp:           array[0..10] of AnsiChar;
  end;
```

GPS_Version: GPS タグのバージョンを示します。(例:"2000")
MapDatum: 測位に用いた地図データを示します。 *1
MapDatumLen: MapDatum に含まれる文字列の長さを表します。 *2
Latitude: 緯度の方角を示します。("N":北緯, "S":南緯)
LatitudeD: 緯度 - 度を示します。
緯度 - 度 = **LatitudeD[0] / LatitudeD[1]**
LatitudeM: 緯度 - 分を示します。
緯度 - 分 = **LatitudeM[0] / LatitudeM[1]**
LatitudeS: 緯度 - 秒を示します。
緯度 - 秒 = **LatitudeS[0] / LatitudeS[1]**
Longitude: 経度の方角を示します。("E":東経, "W":西経)
LongitudeD: 経度 - 度を示します。
経度 - 度 = **LongitudeD[0] / LongitudeD[1]**
LongitudeM: 経度 - 分を示します。
経度 - 分 = **LongitudeM[0] / LongitudeM[1]**
LongitudeS: 経度 - 秒を示します。
経度 - 秒 = **LongitudeS[0] / LongitudeS[1]**
AltitudeRef: 高度の基準を示します。基準単位はメートル。 *2
基準が海抜であり高度が海抜よりも高い場合は 0、高度が海抜よりも低い場合は 1。
Altitude: 高度(m)を示します。
高度 = **Altitude[0] / Altitude[1]**
TimeStampH: GPS 時間 - 時間を示します。 *2
GPS 時間 - 時間 = **TimeStampH[0] / TimeStampH[1]**
TimeStampM: GPS 時間 - 分を示します。 *2
GPS 時間 - 分 = **TimeStampM[0] / TimeStampM[1]**
TimeStampS: GPS 時間 - 秒を示します。 *2
GPS 時間 - 秒 = **TimeStampS[0] / TimeStampS[1]**
ProcessingMethod: 測位に使用した方式の名称を示します(例:" CELLID"," WLAN"," GPS"など)。 *2
ProcessingMethodLen: ProcessingMethod に含まれる文字列の長さを表します。 *2
ProcessingMethodID: ProcessingMethod に書かれる文字コードを判別するために、識別コードを 8 バイト固定で記入し、余った領域には NULL(0x00)でパディングします。 *2

文字コード	コード記入方法(8 バイト)	リファレンス
ASCII	0x41,0x53,0x43,0x49,0x49,0x00,0x00,0x00	ITU-T T.50 IA5
JIS	0x4A,0x49,0x53,0x00,0x00,0x00,0x00,0x00	JIS X0208-1990
Unicode	0x55,0x4E,0x49,0x43,0x4F,0x44,0x45,0x00	Unicode Standard
Undefined	0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00	Undefined

※16 進表記のため、Delphi は 0x を$に置き換えてください。

DateStamp: GPS 日付を示します。フォーマットは"YYYY:MM:DD"。 *2

*1 ImageKit8 でサイズが変更されました。ImageKit7 から移行する場合は注意してください。
*2 ImageKit8 で追加されたメンバー変数です。ImageKit7 から移行する場合は注意してください。

INTOP_TAG: 互換性に関する付属情報を表します。

(1)C++Builder

```
  typedef struct {
```

```
    char        InteroperabilityIndex[4];
    char        InteroperabilityVersion[5];
  } INTOP_TAG;
  typedef INTOP_TAG * PTR_INTOP_TAG;
```

(2)Delphi

```
  type
    INTOP_TAG = Record
      InteroperabilityIndex:      array[0..3] of AnsiChar;
      InteroperabilityVersion:    array[0..4] of AnsiChar;
    end;
```

InteroperabilityIndex: 互換性の規則の種類を示します。
"R98" ExifR98 で規定される R98 ファイルおよび Design rule for Camera File system で規定される DCF 基本ファイル
"THM" Design rule for Camera File system で規定される DCF サムネイルファイル
"R03" Design rule for Camera File system で規定される DCF オプションファイル
InteroperabilityVersion: 互換性のバージョンを示します(例:"0100")。

EXIF_INFO: Exif 情報を表します(IKJpegExifInfo で使用します)。

(1)C++Builder

```
  typedef struct {
    IMAGE_TAG       MainImageInfo;
    IMAGE_TAG       ThumbImageInfo;
    EXIF_TAG        ExifInfo;
    GPS_TAG         GPSInfo;
    INTOP_TAG       IntOpeInfo;
  } EXIF_INFO;
  typedef EXIF_INFO * PTR_EXIF_INFO;
```

(2)Delphi

```
  type
    EXIF_INFO = Record
      MainImageInfo:    IMAGE_TAG;
      ThumbImageInfo:   IMAGE_TAG;
      ExifInfo:         EXIF_TAG;
      GPSInfo:          GPS_TAG;
      IntOpeInfo:INTOP_TAG;
    end;
```

MainImageInfo: 主画像情報を示します。
ThumbImageInfo: サムネイル情報を示します。
ExifInfo: Exif 情報を示します。
GPSInfo: GPS 情報を示します。 *1
IntOpeInfo: 互換性に関する付属情報を示します。 *2

*1 ImageKit7 で追加されたメンバー変数です。ImageKit6 から移行する場合は注意してください。
*2 ImageKit8 で追加されたメンバー変数です。ImageKit6/7 から移行する場合は注意してください。

SAVE_PDF_INFO: PDF ファイル作成時に使用します。

(1)C++Builder

```
  typedef struct {
    void *          pdf;
    char            OwnerPassword[65];
    char            UserPassword[65];
    char            Application[65];
    char            Title[129];
    char            Author[129];
```

```
    char                Subject[129];
    char                Keywords[257];
    BOOL                EnablePrint;
    BOOL                EnableEditAll;
    BOOL                EnableCopy;
    BOOL                EnableEdit;
  } SAVE_PDF_INFO;
  typedef SAVE_PDF_INFO * PTR_SAVE_PDF_INFO;
```

(2)Delphi

```
  type
    SAVE_PDF_INFO = Record
      pdf:              Pointer;
      OwnerPassword:    array[0..64] of AnsiChar;
      UserPassword:     array[0..64] of AnsiChar;
      Application:      array[0..64] of AnsiChar;
      Title:            array[0..128] of AnsiChar;
      Author:           array[0..128] of AnsiChar;
      Subject:          array[0..128] of AnsiChar;
      Keywords:         array[0..256] of AnsiChar;
      EnablePrint:      LongBool;
      EnableEditAll:    LongBool;
      EnableCopy:       LongBool;
      EnableEdit:       LongBool;
    end;
```

pdf: PDF 作成時のデータを指すポインタを表します(ImageKit 内で使用します)。
OwnerPassword: オーナーパスワード(文書の変更や印刷を制限するパスワード)を指定します。
UserPassword: ユーザーパスワード(文書を開くパスワード)を指定します。
Application: アプリケーション名を指定します(文書のプロパティ-概要)。
Title: タイトルを指定します(文書のプロパティ-概要)。
Author: 作成者を指定します(文書のプロパティ-概要)。
Subject: サブタイトルを指定します(文書のプロパティ-概要)。
Keywords: キーワードを指定します(文書のプロパティ-概要)。
EnablePrint: 文書の印刷を許可するかどうかを指定します(TRUE:許可、FALSE:不許可)。
EnableEditAll: 文書の全編集を許可するかどうかを指定します(TRUE:許可、FALSE:不許可)。
EnableCopy: 文書内容のコピー・抽出を許可するかどうかを指定します(TRUE:許可、FALSE: 不許可)。
EnableEdit: 文書の編集を許可するかどうかを指定します(TRUE:許可、FALSE:不許可)。

(*1) UserPassword のみの設定は不可です。OwnerPassword または OwnerPassword と UserPassword の 2 つを設定してください。OwnerPassword と UserPassword を設定する場合、別々の値を設定してください(同じ値は不可)。
(*2) EnablePrint, EnableEditAll, EnableCopy,EnableEdit の設定値が有効となるのは OwnerPassword 設定時です。

【ユーザ関数の定義】

【関数書式】
(1)C++Builder
BOOL _stdcall UserProc(short Percent);
(2)Delphi
function UserProc(Percent: Smallint): LongBool; stdcall;

【引数】

名称	内容
Percent	現在処理している%数

【戻り値】
False(0)の場合は実行している処理を終了します。True(0 以外)は処理を継続します。

ユーザ関数は UserProc という名称で説明しておりますが、実際の名称は何を設定されても構いません。

関数名が UserProc の場合は以下のような形式で引数に渡します。
(1)C++Builder　UserProc
(2)Delphi　LONG_PTR(Addr(UserProc)) or LONG_PTR(@UserProc)

IKBmpFileLoad

【機能】

BMP 形式のファイルからラスタイメージをメモリ上に読み込みます。

【関数書式】

(1)C++Builder

HANDLE IKBmpFileLoad(LPCTSTR FileName, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

(2)Delphi

function IKBmpFileLoad(FileName: PChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】

ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】

対応イメージは 1,4,8,16,24,32 ビットカラーです。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義をご覧ください。

IKBmpFileLoadMem

【機能】
BMP 形式の Raw データからラスタイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKBmpFileLoadMem(HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKBmpFileLoadMem(Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	BMP 形式の Raw データ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
対応イメージは 1,4,8,16,24,32 ビットカラーです。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKBmpFileLoad** 関数と同じです。

IKBmpFileSave

【機能】
ラスタイメージを BMP 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKBmpFileSave(LPCTSTR FileName, HANDLE Handle, BOOL Comp, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKBmpFileSave(FileName: PChar; Handle: THandle; Comp: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ラスタイメージのメモリハンドル
Comp	圧縮フラグ(False(0):圧縮しない、True(0 以外):圧縮する) 圧縮対象は 4 ビットカラー、8 ビットカラーのイメージのみ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対応イメージは 1,4,8,16(グレーは除く),24,32 ビットカラーです。
Comp を True に設定して圧縮する場合は Windows ランレングス形式で圧縮を行いますが、その仕様ではイメージによっては圧縮しないファイルより大きくなる場合があります。当コマンドでは、約 1.6 倍以上の大きさになると保存を中止し、戻り値として False(0)を返します。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合
IKBmpFileSave("c:¥¥abc.bmp",・・・・・);
(2)FTP サーバーにファイルを保存する場合
IKBmpFileSave("ftp://www.newtone.co.jp/image/abc.bmp;;;true;user;password",・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無("true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。
(3)HTTP(S)サーバーにファイルを保存する場合
IKBmpFileSave("http://www.newtone.co.jp/image/abc.bmp;;;;user;password",・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略すると HTTP サーバからの処理となります。

【ImageKit7 との違い】
FTP サーバーや HTTP(S)サーバーに直接ファイルを保存することができるようになりました。

IKBmpFileSaveMem

【機能】
ラスタイメージを BMP 形式で Raw データに保存します。

【関数書式】
(1)C++Builder
BOOL IKBmpFileSaveMem(HANDLE InHandle, HANDLE *OutHandle, BOOL Comp, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKBmpFileSaveMem(InHandle: THandle; var OutHandle: THandle; Comp: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	ラスタイメージのメモリハンドル
OutHandle	保存する Raw データ
Comp	圧縮フラグ(False(0):圧縮しない、True(0 以外):圧縮する) 圧縮対象は 4 ビットカラー、8 ビットカラーのイメージのみ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対応イメージは 1,4,8,16(グレーは除く),24,32 ビットカラーです。
Comp を True に設定して圧縮する場合は Windows ランレングス形式で圧縮を行いますが、その仕様ではイメージによっては圧縮しないファイルより大きくなる場合があります。当コマンドでは、約 1.6 倍以上の大きさになると保存を中止し、戻り値として False(0)を返します。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKBmpFileSave** 関数と同じです。

IKCMYKBmpPlaneFileLoad

【機能】

CMYK プレーン毎に保存してある BMP ファイルからラスタイメージをメモリ上に読み込みます。

【関数書式】

(1)C++Builder

```
HANDLE IKCMYKBmpPlaneFileLoad(LPCTSTR CFileName, LPCTSTR MFileName, LPCTSTR YFileName,
LPCTSTR KFileName);
```

(2)Delphi

```
function IKCMYKBmpPlaneFileLoad(CFileName, MFileName, YFileName, KFileName: PChar): THandle;
```

【引数】

名称	内容
CFileName	C プレーンとして読み込むファイル名
MFileName	M プレーンとして読み込むファイル名
YFileName	Y プレーンとして読み込むファイル名
KFileName	K プレーンとして読み込むファイル名

【戻り値】

ラスタイメージのメモリハンドル（実行に失敗した場合は 0 が返されます）。

【解説】

CMYK それぞれのプレーンを表すイメージは 8 ビットグレーで、出力イメージは 24 ビットカラーです。

IKCMYKBmpPlaneFileSave

【機能】
ラスタイメージを CMYK プレーン毎に BMP 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKCMYKBmpPlaneFileSave(LPCTSTR CFileName, LPCTSTR MFileName, LPCTSTR YFileName, LPCTSTR KFileName, HANDLE Handle);
(2)Delphi
function IKCMYKBmpPlaneFileSave(CFileName, MFileName, YFileName, KFileName: PChar; Handle: THandle): LongBool;

【引数】

名称	内容
CFileName	C プレーンとして保存するファイル名
MFileName	M プレーンとして保存するファイル名
YFileName	Y プレーンとして保存するファイル名
KFileName	K プレーンとして保存するファイル名
Handle	ラスタイメージのメモリハンドル

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
保存対象イメージは 24 ビットカラーです。Handle を CMYK のそれぞれのプレーン毎にファイルに保存します。保存されるイメージは 8 ビットグレーとなります。

IKDxfFileLoad

【機能】
DXF 形式のファイルからベクトルイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKDxfFileLoad(LPCTSTR FileName, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKDxfFileLoad(FileName: PChar; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
Width	読み込むイメージの幅(ピクセル)
Height	読み込むイメージの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Width もしくは Height が 0 以下の場合は幅 800 高さ 600 に設定します。
エンティティタイプは、LINE・POINT・CIRCLE・TEXT・3DFACE・POLYLINE・ARC・SOLID・INSERT に対応しています。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

(注意)
読み込まれるイメージの縦横のサイズは、設定した矩形内(Width * Height)で縦横比を考慮した最も大きなサイズとなりますので、設定したサイズ通りにならない場合があります。

IKDxfFileLoadMem

【機能】
DXF 形式の Raw データからベクトルイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKDxfFileLoadMem(HANDLE Handle, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKDxfFileLoadMem(Handle: THandle; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	DXF 形式の Raw データ
Width	読み込むイメージの幅(ピクセル)
Height	読み込むイメージの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Width もしくは Height が 0 以下の場合は幅 800 高さ 600 に設定します。
エンティティタイプは、LINE・POINT・CIRCLE・TEXT・3DFACE・POLYLINE・ARC・SOLID・INSERT に対応しています。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKDxfFileLoad** 関数と同じです。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

(注意)
読み込まれるイメージの縦横のサイズは、設定した矩形内(Width * Height)で縦横比を考慮した最も大きなサイズとなりますので、設定したサイズ通りにならない場合があります。

IKDxfFileSave

【機能】
ベクトルイメージのメモリハンドルを DXF 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKDxfFileSave(LPCTSTR FileName, HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKDxfFileSave(FileName: PChar; Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ベクトルイメージのメモリハンドル
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
エンティティタイプは、LINE・POINT・ELLIPSE・TEXT・POLYLINE・ARC に対応しています。
保存する際の色については、赤・黄色・緑・水色・青・紫・白の 7 色のいずれかに置き換えられます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合
IKDxfFileSave("c:¥¥abc.dxf",・・・・・);
(2)FTP サーバーにファイルを保存する場合
IKDxfFileSave("ftp://www.newtone.co.jp/image/abc.dxf;;;true;user;password",・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無("true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。
(3)HTTP(S)サーバーにファイルを保存する場合
IKDxfFileSave("http://www.newtone.co.jp/image/abc.dxf;;;user;password",・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。（文字列をセミコロンで区切る）
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略すると HTTP サーバからの処理となります。

【ImageKit7 との違い】
FTP サーバーや HTTP(S)サーバーに直接ファイルを保存することができるようになりました。

IKDxfFileSaveMem

【機能】
ベクトルイメージのメモリハンドルを DXF 形式で Raw データに保存します。

【関数書式】
(1)C++Builder
BOOL IKDxfFileSaveMem(HANDLE InHandle, HANDLE *OutHandle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKDxfFileSaveMem(InHandle: THandle; var OutHandle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	ベクトルイメージのメモリハンドル
OutHandle	保存する Raw データ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
エンティティタイプは、LINE・POINT・ELLIPSE・TEXT・POLYLINE・ARC に対応しています。
保存する際の色については、赤・黄色・緑・水色・青・紫・白の 7 色のいずれかに置き換えられます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については「**Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKDxfFileSave** 関数と同じです。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

IKEmfFileLoad

【機能】
EMF 形式のファイルからベクトルイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKEmfFileLoad(LPCTSTR FileName, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKEmfFileLoad(FileName: PChar; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
Width	読み込むイメージの幅(ピクセル)
Height	読み込むイメージの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Width もしくは Height が 0 以下の場合は EMF ファイルに格納されているサイズで読み込みます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

(注意)
読み込まれるイメージの縦横のサイズは、設定した矩形内(Width * Height)で縦横比を考慮した最も大きなサイズとなりますので、設定したサイズ通りにならない場合があります。

IKEmfFileLoadMem

【機能】
EMF 形式の Raw データからベクトルイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKEmfFileLoadMem(HANDLE Handle, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKEmfFileLoadMem(Handle: THandle; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	EMF 形式の Raw データ
Width	読み込むイメージの幅(ピクセル)
Height	読み込むイメージの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Width もしくは Height が 0 以下の場合は EMF 形式の Raw データに格納されているサイズで読み込みます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKEmfFileLoad** 関数と同じです。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

(注意)
読み込まれるイメージの縦横のサイズは、設定した矩形内(Width * Height)で縦横比を考慮した最も大きなサイズとなりますので、設定したサイズ通りにならない場合があります。

IKEmfFileSave

【機能】
ベクトルイメージのメモリハンドルを EMF 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKEmfFileSave(LPCTSTR FileName, HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKEmfFileSave(FileName: PChar; Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ベクトルイメージのメモリハンドル
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合
IKEmfFileSave("c:¥¥abc.emf",・・・・・);
(2)FTP サーバーにファイルを保存する場合
IKEmfFileSave("ftp://www.newtone.co.jp/image/abc.emf;;;true;user;password",・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無("true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。
(3)HTTP(S)サーバーにファイルを保存する場合
IKEmfFileSave("http://www.newtone.co.jp/image/abc.emf;;;;user;password",・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。(文字列をセミコロンで

区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略するとHTTPサーバからの処理となります。

【ImageKit7との違い】
FTPサーバーやHTTP(S)サーバーに直接ファイルを保存することができるようになりました。

IKEmfFileSaveMem

【機能】
ベクトルイメージのメモリハンドルを EMF 形式で Raw データに保存します。

【関数書式】
(1)C++Builder
BOOL IKEmfFileSaveMem(HANDLE InHandle, HANDLE *OutHandle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKEmfFileSaveMem(InHandle: THandle; var OutHandle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	ベクトルイメージのメモリハンドル
OutHandle	保存する Raw データ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKEmfFileSave** 関数と同じです。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

IKFileLoad

【機能】
イメージファイルからイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKFileLoad(LPCTSTR FileName, int Page, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKFileLoad(FileName: PChar; Page: Integer; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
Page	読み込むファイルのページ番号(0～)
Width	読み込むイメージの幅(ピクセル)
Height	読み込むイメージの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ラスタイメージまたはベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
BMP,DXF,EMF,FPX,GIF,JPEG,JPEG200,PCX,PNG,SXF,SVG,TIFF,WMF などの各ファイル形式を自動的に判断してメモリ上に読み込みます。**ただし、TIFF の JPEG 形式については読み込みに失敗するファイルも存在しますのでご了承ください。**
複数のイメージが存在する場合や FPX のように解像度単位でイメージを持つ場合は、Page で指定されたイメージを読み込みます。1 イメージしかないファイルは Page に指定した値は無効となります。
ラスタイメージを読む込む場合には、Width と Height は無効ですが(何の値でも可)、ベクトルイメージを読み込む場合には Width と Height の値が有効になります。Width もしくは Height が 0 以下の場合、DXF,SXF,WMF の場合は幅 800 高さ 600 に、EMF と SVG の場合はファイルに格納されているサイズとなります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ベクトルイメージを読む込む場合は、事前に
Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll の **IKVectorGdipStart** 関数を実行する必要があります。

(ベクトルイメージ読み込み時の注意)
読み込まれるイメージの縦横のサイズは、設定した矩形内(Width * Height)で縦横比を考慮した最も大きなサイズとなりますので、設定したサイズ通りにならない場合があります。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに配置されているファイルを直接読み込むことができます。

(1)ローカルドライブやネットワークドライブのファイルを読み込む場合
IKFileLoad("c:¥¥abc.jpg",・・・・・);
(2)FTP サーバーに配置されているファイルを読み込む場合
IKFileLoad("ftp://www.newtone.co.jp/image/abc.jpg;;;true;user;password",・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ（PASV）モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。（文字列をセミコロンで区切る）
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無（"true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。

(3)HTTP(S)サーバーに配置されているファイルを読み込む場合

IKFileLoad("http://www.newtone.co.jp/image/abc.jpg;;;;user;password",・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。（文字列をセミコロンで区切る）
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略すると HTTP サーバからの処理となります。

【ImageKit5 との違い】

関数名	**引数の並び**
IK5FileLoad:	FileName, Page, FileUserProc, Caption, Message, Button
IKFileLoad:	FileName, Page, Width, Height, UserProc, Caption, Message, Button

Width と Height が引数に追加されていますが、ラスタイメージをメモリ上に読む込む場合は、Width と Height は無視されますので、何の値を与えても IK5 と同様に動作します。
※FileUserProc と UserProc は同じユーザ関数を表します。

【ImageKit7 との違い】
FTP サーバーや HTTP(S)サーバーに配置されているファイルを直接読み込みできるようになりました。

IKFileLoadAsRawData

【機能】
イメージファイルからそのままの状態でメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKFileLoadAsRawData(LPCTSTR FileName, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKFileLoadAsRawData(FileName: PChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
Raw データのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
読み込んだデータはファイルに保存されているデータと同じ構造です。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ベクトルイメージを読む込む場合は、事前に
Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll の **IKVectorGdipStart** 関数を実行する必要があります。

IKFileLoadMem

【機能】
Raw データからイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKFileLoadMem(HANDLE Handle, int Page, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKFileLoadMem(Handle: THandle; Page: Integer; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	読み込む Raw データ
Page	読み込むファイルのページ番号(0～)
Width	読み込むイメージの幅(ピクセル)
Height	読み込むイメージの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ラスタイメージまたはベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
BMP,DXF,EMF,FPX,GIF,JPEG,JPEG200,PCX,PNG,SXF,SVG,TIFF,WMF などの各イメージ形式を自動的に判断してメモリ上に読み込みます。**ただし、TIFF の JPEG 形式については読み込みに失敗する場合がありますのでご了承ください。**複数のイメージが存在する場合や FPX のように解像度単位でイメージを持つ場合は、Page で指定されたイメージを読み込みます。1 イメージしかない Raw データは Page に指定した値は無効となります。
ラスタイメージを読む込む場合には、Width と Height は無効ですが(何の値でも可)、ベクトルイメージを読み込む場合には Width と Height の値が有効になります。Width もしくは Height が 0 以下の場合、DXF,SXF,WMF の場合は幅 800 高さ 600 に、EMF と SVG の場合は Raw データに格納されているサイズとなります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKFileLoad** 関数と同じです。
ベクトルイメージを読む込む場合は、事前に
Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll の **IKVectorGdipStart** 関数を実行する必要があります。

(ベクトルイメージ読み込み時の注意)
読み込まれるイメージの縦横のサイズは、設定した矩形内(Width * Height)で縦横比を考慮した最も大きなサイズとなりますので、設定したサイズ通りにならない場合があります。

IKFileSaveAsRawData

【機能】
Raw データをファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKFileSaveAsRawData(LPCTSTR FileName, HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKFileSaveAsRawData(FileName: PChar; Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	保存する Raw データ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ベクトルイメージを保存する場合は、事前に
Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll の **IKVectorGdipStart** 関数を実行する必要があります。

IKFileType

【機能】
ファイルからイメージの各種情報を取得します。

【関数書式】
(1)C++Builder
BOOL IKFileType(LPCTSTR FileName, int Page, PTR_IKFILE_INFO FileType);
(2)Delphi
function IKFileType(FileName: PChar; Page: Integer; var FileType: IKFILE_INFO): LongBool;

【引数】

名称	内容
FileName	情報を取得するファイル名
Page	ファイルのページ番号(0～)
FileType	取得するファイル情報の構造体(ユーザ定義型)変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
成功した場合、FileType にファイル情報が設定されます。
対応しているファイル形式であっても、ファイルに該当する情報が保存されていない場合は FileType のメンバー変数に 0 もしくは空文字列が設定されます。
IKFILE_INFO については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」の IKFILE_INFO のメンバー変数の説明**をご覧ください。

【ImageKit6 との違い】
IKFILE_INFO 構造体にメンバー変数が追加されました。

IKFileTypeMem

【機能】
Raw データからイメージの各種情報を取得します。

【関数書式】
(1)C++Builder
BOOL IKFileTypeMem(HANDLE Handle, int Page, PTR_IKFILE_INFO FileType);
(2)Delphi
function IKFileTypeMem(Handle: THandle; Page: Integer; var FileType: IKFILE_INFO): LongBool;

【引数】

名称	内容
Handle	情報を取得する Raw データ
Page	ファイルのページ番号(0～)
FileType	取得するファイル情報の構造体(ユーザ定義型)変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
成功した場合、FileType にファイル情報が設定されます。
対応しているファイル形式であっても、Raw データに該当する情報が保存されていない場合は FileType のメンバー変数に 0 もしくは空文字列が設定されます。ファイルの作成日時・アクセス日時・更新日時を示すメンバー変数には 0 が設定されます。
IKFILE_INFO については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」の IKFILE_INFO のメンバー変数の説明**をご覧ください。
ファイルあるいは Raw データから情報を取得する違いはありますが、動作としては **IKFileType** 関数と同じです。

IKFpxFileLoad

【機能】

FPX 形式のファイルからラスタイメージをメモリ上に読み込みます。

【関数書式】

(1)C++Builder

HANDLE IKFpxFileLoad(LPCTSTR FileName, int Page, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

(2)Delphi

function IKFpxFileLoad(FileName: PChar; Page: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
Page	読み込むファイルのページ(解像度)番号(0～) 0 がオリジナルイメージで、1 以降が一つ前のイメージの縦横の解像度をそれぞれ半分にしたものになります。最後のページは 64x64 以下のイメージとなります。
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】

ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】

Page で指定されたイメージをメモリ上に読み込みます。対応イメージは 8 ビットグレースケールと 24 ビットカラーです。イメージファイルのページ数は **IKFileType** 関数で取得できます。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

＜FlashPix について＞

FlashPix の最大の特徴は、マルチ・レゾリューション(複数解像度)で、さらにタイル構造を持つファイル・フォーマットであり、基画像から最小画像までの、画素数の異なる複数画面を一つのファイルの中に持つことができ、編集・印刷など異なった用途に最適な解像度で対応することができます。それぞれの解像度の画面は、64x64 のブロック(タイル)に分割されています。タイル内は、非圧縮・JPEG 圧縮・単色圧縮のいずれかが可能です。

IKFpxFileLoadMem

【機能】
FPX 形式の Raw データからラスタイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKFpxFileLoadMem(HANDLE Handle, int Page, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKFpxFileLoadMem(Handle: THandle; Page: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	FPX 形式の Raw データ
Page	読み込むファイルのページ(解像度)番号(0～) 0 がオリジナルイメージで、1 以降が一つ前のイメージの縦横の解像度をそれぞれ半分にしたものになります。最後のページは 64x64 以下のイメージとなります。
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Page で指定されたイメージをメモリ上に読み込みます。対応イメージは 8 ビットグレースケールと 24 ビットカラーです。イメージファイルのページ数は **IKFileType** 関数で取得できます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKFpxFileLoad** 関数と同じです。

(注意)
内部で一時的にテンポラリファイルを作成します。

IKFpxFileSave

【機能】
ラスタイメージを FPX 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKFpxFileSave(LPCTSTR FileName, HANDLE Handle, int Comp, int Quality, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKFpxFileSave(FileName: PChar; Handle: THandle; Comp, Quality: Integer; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ラスタイメージのメモリハンドル
Comp	圧縮フラグ(0:非圧縮 1:Single Color 2:JPEG) Single Color は全て同じ RGB のピクセルで構成されているイメージが対象となります。 JPEG は JPEG 形式で圧縮します。
Quality	JPEG 圧縮品質係数(0～100/推奨 75) ※Comp が 2 の時有効となります。
Comment	保存するコメント文字列(MAX:1023 バイト) 次のように設定してください。「Title + 0x0d + Subject + ・・・ + 0x0d」 Title: タイトル Subject: サブタイトル Author: 作成者 Comment: コメント ※16 進表記のため、Delphi は 0x を$に置き換えてください。
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対応イメージは 8 ビットグレースケールと 24 ビットカラーです。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合
IKFpxFileSave(″c:¥¥abc.fpx″,・・・・・);
(2)FTP サーバーにファイルを保存する場合
IKFpxFileSave(″ftp://www.newtone.co.jp/image/abc.fpx;;;true;user;password″,・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合はプロパティに「FTP サーバの名

称もしくはIPアドレス;プロキシサーバの名称またはIPアドレス;ポート番号;パッシブモード接続の有無」を渡してください。（文字列をセミコロンで区切る）
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無（"true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。

(3)HTTP(S)サーバーにファイルを保存する場合

IKFpxFileSave("http://www.newtone.co.jp/image/abc.fpx;;;;user;password",・・・・・);

(**)FileNameを設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、またはHTTPSを利用する場合はプロパティに「HTTP(S)サーバの名称もしくはIPアドレス;プロキシサーバの名称またはIPアドレス;ポート番号;HTTPS」を渡してください。（文字列をセミコロンで区切る）
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略するとHTTPサーバからの処理となります。

【ImageKit7との違い】

FTPサーバーやHTTP(S)サーバーに直接ファイルを保存することができるようになりました。

IKFpxFileSaveMem

【機能】
ラスタイメージを FPX 形式で Raw データに保存します。

【関数書式】
(1)C++Builder
BOOL IKFpxFileSaveMem(HANDLE InHandle, HANDLE *OutHandle, int Comp, int Quality, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKFpxFileSaveMem(InHandle: THandle; var OutHandle: THandle; Comp, Quality: Integer; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	ラスタイメージのメモリハンドル
OutHandle	保存する Raw データ
Comp	圧縮フラグ(0:非圧縮 1:Single Color 2:JPEG) Single Color は全て同じ RGB のピクセルで構成されているイメージが対象となります。 JPEG は JPEG 形式で圧縮します。
Quality	JPEG 圧縮品質係数(0～100/推奨 75) ※Comp が 2 の時有効となります。
Comment	保存するコメント文字列(MAX:1023 バイト) 次のように設定してください。「Title + 0x0d + Subject + ・・・ + 0x0d」 Title: タイトル Subject: サブタイトル Author: 作成者 Comment: コメント ※16 進表記のため、Delphi は 0x を$に置き換えてください。
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対応イメージは 8 ビットグレースケールと 24 ビットカラーです。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKFpxFileSave** 関数と同じです。

(注意)
内部で一時的にテンポラリファイルを作成します。

IKFTPConnect

【機能】
FTP サーバに接続します。

【関数書式】
(1)C++Builder
BOOL IKFTPConnect(LPCTSTR ServerName, LPCTSTR UserName, LPCTSTR Password, HINTERNET *hOpen, HINTERNET *hConnect);
(2)Delphi
function IKFTPConnect(ServerName, UserName, Password: LPCTSTR; hOpen: PHINTERNET; hConnect: PHINTERNET): LongBool;

【引数】

名称	内容
ServerName	FTP サーバの名称もしくは IP アドレス (*1)
UserName	ユーザ名
Password	パスワード
hOpen	FTP サーバへのオープンハンドル
hConnect	FTP サーバへのセッションハンドル

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。
当関数で FTP サーバへ接続し、IKFTPDeleteFileEx,IKFTPGetFileEx,IKFTPPutFileEx,IKFTPRenameFileEx 関数を使用します。

(*1)
プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合は、引数の ServerName に「FTP サーバの名称もしくは IP アドレス;プロキシサーバの IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無を省略するとパッシブモード接続となります。

A.プロキシサーバを設定する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx"
B.ポート番号を 22 に変更する例:
ServerName = "www.newtone.co.jp;;22"
C.パッシブモードではなくアクティブモードで接続する例:
ServerName = "www.newtone.co.jp;;;FALSE"
D.プロキシサーバを設定し、ポート番号を 22 に変更する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx;22"
E.プロキシサーバを設定し、ポート番号を 22 に変更し、アクティブモードで接続する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx;22;FALSE"

IKFTPDeleteFile

【機能】
FTP サーバに存在するファイルを削除します。

【関数書式】
(1)C++Builder
BOOL IKFTPDeleteFile(LPCTSTR ServerName, LPCTSTR DeleteFile, LPCTSTR UserName, LPCTSTR Password);
(2)Delphi
function IKFTPDeleteFile(ServerName, DeleteFile, UserName, Password: PChar): LongBool;

【引数】

名称	内容
ServerName	FTP サーバの名称もしくは IP アドレス (*1)
DeleteFile	削除するファイル名
UserName	ユーザ名
Password	パスワード

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。当関数では FTP サーバへの接続と切断を行います。

(*1)
プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合は、引数の ServerName に「FTP サーバの名称もしくは IP アドレス;プロキシサーバの IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無を省略するとパッシブモード接続となります。

A.プロキシサーバを設定する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx"
B.ポート番号を 22 に変更する例:
ServerName = "www.newtone.co.jp;;22"
C.パッシブモードではなくアクティブモードで接続する例:
ServerName = "www.newtone.co.jp;;;FALSE"
D.プロキシサーバを設定し、ポート番号を 22 に変更する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx;22"
E.プロキシサーバを設定し、ポート番号を 22 に変更し、アクティブモードで接続する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx;22;FALSE"

IKFTPDeleteFileEx

【機能】
FTP サーバに存在するファイルを削除します。

【関数書式】
(1)C++Builder
BOOL IKFTPDeleteFileEx(HINTERNET hConnect, LPCTSTR DeleteFile);
(2)Delphi
function IKFTPDeleteFileEx(hConnect: HINTERNET; DeleteFile: LPCTSTR): LongBool;

【引数】

名称	内容
hConnect	IKFTPConnect 関数で取得したセッションハンドル
DeleteFile	削除するファイル名

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。IKFTPConnect 関数で FTP サーバに接続してから使用します。

コード例:
(1)C++Builder
```
HINTERNET hOpen, hConnect;
if (IKFTPConnect("www.newtone.co.jp", "User", "Password", &hOpen, &hConnect) == FALSE) return;
IKFTPDeleteFileEx(hConnect, "images/001.jpg");
IKFTPDisconnect(hOpen, hConnect);
```
(2)Delphi
```
hOpen, hConnect: HINTERNET;
if (IKFTPConnect('www.newtone.co.jp', 'User', 'Password', @hOpen, @hConnect) = False) then Exit;
IKFTPDeleteFileEx(hConnect, 'images/001.jpg');
IKFTPDisconnect(hOpen, hConnect);
```

IKFTPDisconnect

【機能】
接続している FTP サーバを切断します。

【関数書式】
(1)C++Builder
BOOL IKFTPDisconnect(HINTERNET hOpen, HINTERNET hConnect);
(2)Delphi
function IKFTPDisconnect(hOpen, hConnect: HINTERNET): LongBool;

【引数】

名称	内容
hOpen	IKFTPConnect 関数で取得したオープンハンドル
hConnect	IKFTPConnect 関数で取得したセッションハンドル

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。IKFTPConnect 関数で接続した FTP サーバを切断します。

IKFTPGetFile

【機能】
FTP サーバからファイルを取得します。

【関数書式】
(1)C++Builder
BOOL IKFTPGetFile(LPCTSTR ServerName, LPCTSTR RemoteFile, LPCTSTR LocalFile, LPCTSTR UserName, LPCTSTR Password, long TransPercent, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKFTPGetFile(ServerName, RemoteFile, LocalFile, UserName, Password: PChar; TransPercent: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
ServerName	FTP サーバの名称もしくは IP アドレス (*1)
RemoteFile	FTP サーバから取得するファイル名
LocalFile	ローカルに保存するファイル名
UserName	ユーザ名
Password	パスワード
TransPercent	転送単位(1～100)%
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。当関数では FTP サーバへの接続と切断を行います。TransPercent には転送単位を%で与えます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

(*1)
プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合は、引数の ServerName に「FTP サーバの名称もしくは IP アドレス;プロキシサーバの IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無を省略するとパッシブモード接続となります。
A.プロキシサーバを設定する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx"
B.ポート番号を 22 に変更する例:
ServerName = "www.newtone.co.jp;;22"
C.パッシブモードではなくアクティブモードで接続する例:
ServerName = "www.newtone.co.jp;;;FALSE"
D.プロキシサーバを設定し、ポート番号を 22 に変更する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx;22"
E.プロキシサーバを設定し、ポート番号を 22 に変更し、アクティブモードで接続する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx;22;FALSE"

IKFTPGetFileEx

【機能】
FTP サーバからファイルを取得します。

【関数書式】
(1)C++Builder
```
BOOL IKFTPGetFileEx(HINTERNET hConnect, LPCTSTR RemoteFile, LPCTSTR LocalFile, long TransPercent,
IKPROCESSPROC UserProc);
```
(2)Delphi
```
function IKFTPGetFileEx(hConnect: HINTERNET; RemoteFile, LocalFile: LPCTSTR; TransPercent: Longint;
UserProc: LONG_PTR): LongBool;
```

【引数】

名称	内容
hConnect	IKFTPConnect 関数で取得したセッションハンドル
RemoteFile	FTP サーバから取得するファイル名
LocalFile	ローカルに保存するファイル名
TransPercent	転送単位(1～100)%
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。IKFTPConnect 関数で FTP サーバに接続してから使用します。
TransPercent には転送単位を%で与えます。

コード例:
(1)C++Builder
```
HINTERNET hOpen, hConnect;
if (IKFTPConnect("www.newtone.co.jp", "User", "Password", &hOpen, &hConnect) == FALSE) return;
IKFTPGetFileEx(hConnect, "images/001.jpg", "C:¥¥Images¥¥001.jpg", 10, 0);
IKFTPDisconnect(hOpen, hConnect);
```
(2)Delphi
```
hOpen, hConnect: HINTERNET;
if (IKFTPConnect('www.newtone.co.jp', 'User', 'Password', @hOpen, @hConnect) = False) then Exit;
IKFTPGetFileEx(hConnect, 'images/001.jpg', 'C:¥Images¥001.jpg', 10, 0);
IKFTPDisconnect(hOpen, hConnect);
```

IKFTPPutFile

【機能】
FTP サーバへファイルを転送します。

【関数書式】
(1)C++Builder
BOOL IKFTPPutFile(LPCTSTR ServerName, LPCTSTR RemoteFile, LPCTSTR LocalFile, LPCTSTR UserName, LPCTSTR Password, long TransPercent, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKFTPPutFile(ServerName, RemoteFile, LocalFile, UserName, Password: PChar; TransPercent: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
ServerName	FTP サーバの名称もしくは IP アドレス (*1)
RemoteFile	転送先(FTP サーバ)ファイル名
LocalFile	転送元(ローカル)ファイル名
UserName	ユーザ名
Password	パスワード
TransPercent	転送単位(1～100)%
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。当関数では FTP サーバへの接続と切断を行います。TransPercent には転送単位を%で与えます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

(*1)
プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合は、引数の ServerName に「FTP サーバの名称もしくは IP アドレス;プロキシサーバの IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無を省略するとパッシブモード接続となります。
A.プロキシサーバを設定する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx"
B.ポート番号を 22 に変更する例:
ServerName = "www.newtone.co.jp;;22"
C.パッシブモードではなくアクティブモードで接続する例:
ServerName = "www.newtone.co.jp;;;FALSE"
D.プロキシサーバを設定し、ポート番号を 22 に変更する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx;22"
E.プロキシサーバを設定し、ポート番号を 22 に変更し、アクティブモードで接続する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx;22;FALSE"

IKFTPPutFileEx

【機能】
FTP サーバへファイルを転送します。

【関数書式】
(1)C++Builder
BOOL IKFTPPutFileEx(HINTERNET hConnect, LPCTSTR RemoteFile, LPCTSTR LocalFile, long TransPercent, IKPROCESSPROC UserProc);
(2)Delphi
function IKFTPPutFileEx(hConnect: HINTERNET; RemoteFile, LocalFile: LPCTSTR; TransPercent: Longint; UserProc: LONG_PTR): LongBool;

【引数】

名称	内容
hConnect	IKFTPConnect 関数で取得したセッションハンドル
RemoteFile	転送先(FTP サーバ)ファイル名
LocalFile	転送元(ローカル)ファイル名
TransPercent	転送単位(1～100)%
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。IKFTPConnect 関数で FTP サーバに接続してから使用します。
TransPercent には転送単位を%で与えます。

コード例:
(1)C++Builder
```
HINTERNET hOpen, hConnect;
if (IKFTPConnect("www.newtone.co.jp", "User", "Password", &hOpen, &hConnect) == FALSE) return;
IKFTPPutFileEx(hConnect, "images/001.jpg", "C:¥¥Images¥¥001.jpg", 10, 0);
IKFTPDisconnect(hOpen, hConnect);
```
(2)Delphi
```
hOpen, hConnect: HINTERNET;
if (IKFTPConnect('www.newtone.co.jp', 'User', 'Password', @hOpen, @hConnect) = False) then Exit;
IKFTPPutFileEx(hConnect, 'images/001.jpg', 'C:¥Images¥001.jpg', 10, 0);
IKFTPDisconnect(hOpen, hConnect);
```

IKFTPRenameFile

【機能】
FTP サーバに存在するファイルの名称を変更します。

【関数書式】
(1)C++Builder
BOOL IKFTPRenameFile(LPCTSTR ServerName, LPCTSTR OldFile, LPCTSTR NewFile, LPCTSTR UserName, LPCTSTR Password);
(2)Delphi
function IKFTPRenameFile(ServerName, OldFile, NewFile, UserName, Password: PChar): LongBool;

【引数】

名称	内容
ServerName	FTP サーバの名称もしくは IP アドレス (*1)
OldFile	変更前のファイル名
NewFile	変更後のファイル名
UserName	ユーザ名
Password	パスワード

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。当関数では FTP サーバへの接続と切断を行います。

(*1)
プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合は、引数の ServerName に「FTP サーバの名称もしくは IP アドレス;プロキシサーバの IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無を省略するとパッシブモード接続となります。

A.プロキシサーバを設定する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx"
B.ポート番号を 22 に変更する例:
ServerName = "www.newtone.co.jp;;22"
C.パッシブモードではなくアクティブモードで接続する例:
ServerName = "www.newtone.co.jp;;;FALSE"
D.プロキシサーバを設定し、ポート番号を 22 に変更する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx;22"
E.プロキシサーバを設定し、ポート番号を 22 に変更し、アクティブモードで接続する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx;22;FALSE"

IKFTPRenameFileEx

【機能】
FTP サーバに存在するファイルの名称を変更します。

【関数書式】
(1)C++Builder
BOOL IKFTPRenameFileEx(HINTERNET hConnect, LPCTSTR OldFile, LPCTSTR NewFile);
(2)Delphi
function IKFTPRenameFileEx(hConnect: HINTERNET; OldFile, NewFile: LPCTSTR): LongBool;

【引数】

名称	内容
hConnect	IKFTPConnect 関数で取得したセッションハンドル
OldFile	変更前のファイル名
NewFile	変更後のファイル名

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。IKFTPConnect 関数で FTP サーバに接続してから使用します。

コード例:
(1)C++Builder

```
HINTERNET hOpen, hConnect;
if (IKFTPConnect("www.newtone.co.jp", "User", "Password", &hOpen, &hConnect) == FALSE) return;
IKFTPRenameFileEx(hConnect, "images/001.jpg", "002.jpg");
IKFTPDisconnect(hOpen, hConnect);
```

(2)Delphi

```
hOpen, hConnect: HINTERNET;
if (IKFTPConnect('www.newtone.co.jp', 'User', 'Password', @hOpen, @hConnect) = False) then Exit;
IKFTPRenameFileEx(hConnect, 'images/001.jpg', '002.jpg');
IKFTPDisconnect(hOpen, hConnect);
```

IKGifFileLoad

【機能】

GIF 形式のファイルからラスタイメージをメモリ上に読み込みます。

【関数書式】

(1)C++Builder

HANDLE IKGifFileLoad(LPCTSTR FileName, int Page, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

(2)Delphi

function IKGifFileLoad(FileName: PChar; Page: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
Page	読み込むファイルのページ番号(0～)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】

ラスタイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】

複数のイメージが存在する場合は、Page で指定されたイメージを読み込みます。1 イメージしかないファイルは Page に指定した値は無効となります。対応イメージは 1,4,8 ビットカラーです。

イメージファイルのページ数は **IKFileType** 関数で取得できます。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

IKGifFileLoadMem

【機能】
GIF 形式の Raw データからラスタイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKGifFileLoadMem(HANDLE Handle, int Page, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKGifFileLoadMem(Handle: THandle; Page: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	GIF 形式の Raw データ
Page	読み込むファイルのページ番号(0～)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ラスタイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
複数のイメージが存在する場合は、Page で指定されたイメージを読み込みます。1 イメージしかない Raw データは Page に指定した値は無効となります。対応イメージは 1,4,8 ビットカラーです。
イメージファイルのページ数は **IKFileTypeMem** 関数で取得できます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKGifFileLoad** 関数と同じです。

IKGifFileSave

【機能】
ラスタイメージを GIF 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKGifFileSave(LPCTSTR FileName, HANDLE Handle, BOOL Interlace, BOOL Trans, BYTE Red, BYTE Green, BYTE Blue, BOOL Anime, int Delay, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKGifFileSave(FileName: PChar; Handle: THandle; Interlace, Trans: LongBool; Red, Green, Blue: Byte; Anime: LongBool; Delay: Integer; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ラスタイメージのメモリハンドル
Interlace	インタレース(False(0):設定なし、True(0 以外):設定あり)
Trans	透過(False(0):設定なし、True(0 以外):設定あり)
Red	透過パレットの赤 (0～255) ※Trans が True(0 以外)の場合に有効となります。
Green	透過パレットの緑 (0～255) ※Trans が True(0 以外)の場合に有効となります。
Blue	透過パレットの青 (0～255) ※Trans が True(0 以外)の場合に有効となります。
Anime	False(0):シングルイメージ、True(0 以外):アニメーション(マルチイメージ) True(0 以外)を設定するとファイルの最後のページに該当するイメージを付加します。
Delay	画像を表示した後、次の画像を表示するまでの時間(1/100 秒単位) ※Anime が True(0 以外)の場合に有効です。
Comment	保存するコメント文字列(MAX:1023 バイト)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対応イメージは 1,4,8 ビットカラーです。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合
IKGifFileSave(″c:¥¥abc.gif″,・・・・・);
(2)FTP サーバーにファイルを保存する場合
IKGifFileSave(″ftp://www.newtone.co.jp/image/abc.gif;;;true;user;password″,・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ（PASV）モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。（文字列をセミコロンで区切る）
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無（"true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。

(3)HTTP(S)サーバーにファイルを保存する場合

IKGifFileSave("http://www.newtone.co.jp/image/abc.gif;;;;user;password",・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。（文字列をセミコロンで区切る）
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略すると HTTP サーバからの処理となります。

【ImageKit7 との違い】

FTP サーバーや HTTP(S)サーバーに直接ファイルを保存することができるようになりました。

IKGifFileSaveMem

【機能】
ラスタイメージを GIF 形式で Raw データに保存します。

【関数書式】
(1)C++Builder
BOOL IKGifFileSaveMem(HANDLE InHandle, HANDLE *OutHandle, BOOL Interlace, BOOL Trans, BYTE Red, BYTE Green, BYTE Blue, BOOL Anime, int Delay, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKGifFileSaveMem(InHandle: THandle; var OutHandle: THandle; Interlace, Trans: LongBool; Red, Green, Blue: Byte; Anime: LongBool; Delay: Integer; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	ラスタイメージのメモリハンドル
OutHandle	保存する Raw データ
Interlace	インタレース(False(0):設定なし、True(0 以外):設定あり)
Trans	透過(False(0):設定なし、True(0 以外):設定あり)
Red	透過パレットの赤 (0～255) ※Trans が True(0 以外)の場合に有効となります。
Green	透過パレットの緑 (0～255) ※Trans が True(0 以外)の場合に有効となります。
Blue	透過パレットの青 (0～255) ※Trans が True(0 以外)の場合に有効となります。
Anime	False(0):シングルイメージ、True(0 以外):アニメーション(マルチイメージ) True(0 以外)を設定するとファイルの最後のページに該当するイメージを付加します。
Delay	画像を表示した後、次の画像を表示するまでの時間(1/100 秒単位) ※Anime が True(0 以外)の場合に有効です。
Comment	保存するコメント文字列(MAX:1023 バイト)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対応イメージは 1,4,8 ビットカラーです。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKGifFileSave** 関数と同じです。

IKHTTPConnect

【機能】
HTTP(S)サーバに接続します。

【関数書式】
(1)C++Builder
BOOL IKHTTPConnect(LPCTSTR ServerName, LPCTSTR UserName, LPCTSTR Password, HINTERNET *hOpen, HINTERNET *hConnect);
(2)Delphi
function IKHTTPConnect(ServerName, UserName, Password: LPCTSTR; hOpen: PHINTERNET; hConnect: PHINTERNET): LongBool;

【引数】

名称	内容
ServerName	HTTP(S)サーバの名称もしくは IP アドレス (*1)
UserName	ユーザ名
Password	パスワード
hOpen	HTTP(S)サーバへのオープンハンドル
hConnect	HTTP(S)サーバへのセッションハンドル

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。
当関数で HTTP(S)サーバへ接続し、IKHTTPGetFileEx,IKHTTPPutFileEx 関数を使用します。

(*1)
プロキシサーバを設定したりポート番号を変更、またはHTTPSを利用する場合は引数のServerNameに「HTTP(S)サーバの名称もしくはIPアドレス;プロキシサーバのIPアドレス;ポート番号;HTTPS」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略するとHTTPサーバからの処理となります。

A.プロキシサーバを設定する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx"
B.ポート番号を 81 に変更する例:
ServerName = "www.newtone.co.jp;;81"
C.プロキシサーバを設定し、ポート番号を 81 に変更する例:
ServerName = "www.newtone.co.jp;xxx.xxx.x.xxx;81"
D.HTTPS を利用する例:
ServerName = "www.newtone.co.jp;;;HTTPS"
E.HTTPS を利用し、ポート番号を 444 に変更する例:
ServerName = "www.newtone.co.jp;;444;HTTPS"

IKHTTPDisconnect

【機能】
接続している HTTP(S)サーバを切断します。

【関数書式】
(1)C++Builder
BOOL IKHTTPDisconnect(HINTERNET hOpen, HINTERNET hConnect);
(2)Delphi
function IKHTTPDisconnect(hOpen, hConnect: HINTERNET): LongBool;

【引数】

名称	内容
hOpen	IKHTTPConnect 関数で取得したオープンハンドル
hConnect	IKHTTPConnect 関数で取得したセッションハンドル

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。IKHTTPConnect 関数で接続した HTTP(S)サーバを切断します。

IKHTTPGetFile

【機能】
HTTP(S)サーバからファイルを取得します。

【関数書式】
(1)C++Builder
BOOL IKHTTPGetFile(LPCTSTR ServerName, LPCTSTR RemoteFile, LPCTSTR LocalFile, LPCTSTR UserName, LPCTSTR Password, long TransPercent, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKHTTPGetFile(ServerName, RemoteFile, LocalFile, UserName, Password: PChar; TransPercent: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
ServerName	HTTP(S)サーバの名称もしくは IP アドレス (*1)
RemoteFile	HTTP(S)サーバから取得するファイル名
LocalFile	ローカルに保存するファイル名
UserName	ユーザ名
Password	パスワード
TransPercent	転送単位(1～100)%
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。当関数では HTTP(S)サーバへの接続と切断を行います。TransPercent には転送単位を%で与えます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

(*1)
プロキシサーバを設定したりポート番号を変更、またはHTTPSを利用する場合は引数の ServerName に「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの IP アドレス;ポート番号;HTTPS」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略すると HTTP サーバからの処理となります。

A.プロキシサーバを設定する例:
ServerName = “www.newtone.co.jp;xxx.xxx.x.xxx”
B.ポート番号を 81 に変更する例:
ServerName = “www.newtone.co.jp;;81”
C.プロキシサーバを設定し、ポート番号を 81 に変更する例:
ServerName = “www.newtone.co.jp;xxx.xxx.x.xxx;81”
D.HTTPS を利用する例:
ServerName = “www.newtone.co.jp;;;HTTPS”
E.HTTPS を利用し、ポート番号を 444 に変更する例:
ServerName = “www.newtone.co.jp;;444;HTTPS”

IKHTTPGetFileEx

【機能】
HTTP(S)サーバからファイルを取得します。

【関数書式】
(1)C++Builder
BOOL IKHTTPGetFileEx(HINTERNET hConnect, LPCTSTR RemoteFile, LPCTSTR LocalFile, long TransPercent, IKPROCESSPROC UserProc);
(2)Delphi
function IKHTTPGetFileEx(hConnect: HINTERNET; RemoteFile, LocalFile: LPCTSTR; TransPercent: Longint; UserProc: LONG_PTR): LongBool;

【引数】

名称	内容
hConnect	IKHTTPConnect 関数で取得したセッションハンドル
RemoteFile	HTTP(S)サーバから取得するファイル名
LocalFile	ローカルに保存するファイル名
TransPercent	転送単位(1～100)%
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。IKHTTPConnect 関数で HTTP(S)サーバに接続してから使用します。
TransPercent には転送単位を%で与えます。

コード例:
(1)C++Builder
```
HINTERNET hOpen, hConnect;
if (IKHTTPConnect("www.newtone.co.jp", "User", "Password", &hOpen, &hConnect) == FALSE) return;
IKHTTPGetFileEx(hConnect, "images/001.jpg", "C:¥¥Images¥¥001.jpg", 10, 0);
IKHTTPDisconnect(hOpen, hConnect);
```
(2)Delphi
```
hOpen, hConnect: HINTERNET;
if (IKHTTPConnect('www.newtone.co.jp', 'User', 'Password', @hOpen, @hConnect) = False) then Exit;
IKHTTPGetFileEx(hConnect, 'images/001.jpg', 'C:¥Images¥001.jpg', 10, 0);
IKHTTPDisconnect(hOpen, hConnect);
```

IKHTTPPutFile

【機能】
HTTP(S)サーバへファイルを転送します。

【関数書式】
(1)C++Builder
BOOL IKHTTPPutFile(LPCTSTR ServerName, LPCTSTR RemoteFile, LPCTSTR LocalFile, LPCTSTR UserName, LPCTSTR Password, long TransPercent, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKHTTPPutFile(ServerName, RemoteFile, LocalFile, UserName, Password: PChar; TransPercent: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
ServerName	HTTP(S)サーバの名称もしくは IP アドレス (*1)
RemoteFile	転送先(HTTP(S)サーバ)ファイル名
LocalFile	転送元(ローカル)ファイル名
UserName	ユーザ名
Password	パスワード
TransPercent	転送単位(1～100)%
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。当関数では HTTP(S)サーバへの接続と切断を行います。TransPercent には転送単位を%で与えます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

(*1)
プロキシサーバを設定したりポート番号を変更、またはHTTPSを利用する場合は引数の ServerName に「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの IP アドレス;ポート番号;HTTPS」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略すると HTTP サーバへの処理となります。

A.プロキシサーバを設定する例:
ServerName = “www.newtone.co.jp;xxx.xxx.x.xxx”
B.ポート番号を 81 に変更する例:
ServerName = “www.newtone.co.jp;;81”
C.プロキシサーバを設定し、ポート番号を 81 に変更する例:
ServerName = “www.newtone.co.jp;xxx.xxx.x.xxx;81”
D.HTTPS を利用する例:
ServerName = “www.newtone.co.jp;;;HTTPS”
E.HTTPS を利用し、ポート番号を 444 に変更する例:
ServerName = “www.newtone.co.jp;;444;HTTPS”

IKHTTPPutFileEx

【機能】
HTTP(S)サーバへファイルを転送します。

【関数書式】
(1)C++Builder
BOOL IKHTTPPutFileEx(HINTERNET hConnect, LPCTSTR RemoteFile, LPCTSTR LocalFile, long TransPercent, IKPROCESSPROC UserProc);
(2)Delphi
function IKHTTPPutFileEx(hConnect: HINTERNET; RemoteFile, LocalFile: LPCTSTR; TransPercent: Longint; UserProc: LONG_PTR): LongBool;

【引数】

名称	内容
hConnect	IKHTTPConnect 関数で取得したセッションハンドル
RemoteFile	転送先(HTTP(S)サーバ)ファイル名
LocalFile	転送元(ローカル)ファイル名
UserName	ユーザ名
Password	パスワード
TransPercent	転送単位(1～100)%
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
クライアントからの使用に限ります。IKHTTPConnect 関数で HTTP(S)サーバに接続してから使用します。
TransPercent には転送単位を%で与えます。

コード例:
(1)C++Builder
```
HINTERNET hOpen, hConnect;
if (IKHTTPConnect("www.newtone.co.jp", "User", "Password", &hOpen, &hConnect) == FALSE) return;
IKHTTPPutFileEx(hConnect, "images/001.jpg", "C:¥¥Images¥¥001.jpg", 10, 0);
IKHTTPDisconnect(hOpen, hConnect);
```
(2)Delphi
```
hOpen, hConnect: HINTERNET;
if (IKHTTPConnect('www.newtone.co.jp', 'User', 'Password', @hOpen, @hConnect) = False) then Exit;
IKHTTPPutFileEx(hConnect, 'images/001.jpg', 'C:¥Images¥001.jpg', 10, 0);
IKHTTPDisconnect(hOpen, hConnect);
```

IKJ2kFileLoad

【機能】

JPEG2000 形式のファイルからラスタイメージをメモリ上に読み込みます。

【関数書式】

(1)C++Builder

```
HANDLE IKJ2kFileLoad(LPCTSTR FileName, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message,
LPCTSTR Button);
```

(2)Delphi

```
function IKJ2kFileLoad(FileName: PChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;
```

【引数】

名称	内容
FileName	読み込むファイル名
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】

ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】

対応イメージは 8 ビットグレースケール、24 ビットカラーです。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

IKJ2kFileLoadMem

【機能】

JPEG2000 形式の Raw データからラスタイメージをメモリ上に読み込みます。

【関数書式】

(1)C++Builder

HANDLE IKJ2kFileLoadMem(HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

(2)Delphi

function IKJ2kFileLoadMem(Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	JPEG2000 形式の Raw データ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】

ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】

対応イメージは 8 ビットグレースケール、24 ビットカラーです。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKJ2kFileLoad** 関数と同じです。

IKJ2kFileSave

【機能】
ラスタイメージを JPEG2000 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKJ2kFileSave(LPCTSTR FileName, HANDLE Handle, BOOL JP2kFormat, BOOL Reversible, double Size, long TileWidth, long TileHeight, short NumrResLevel, short PrecinctWidth, short PrecinctHeight, short CodeBlockWidth, short CodeBlockHeight, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKJ2kFileSave(FileName: PChar; Handle: THandle; JP2kFormat, Reversible: LongBool; Size: Double; TileWidth, TileHeight: Longint; NumrResLevel, PrecinctWidth, PrecinctHeight, CodeBlockWidth, CodeBlockHeight: Smallint; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ラスタイメージのメモリハンドル
JP2kFormat	フォーマット(False(0):JPEG2000 Part1、True(0 以外):JPEG2000 Code Stream)
Reversible	圧縮方法(False(0):非可逆、True(0 以外):可逆)
Size	ファイルの保存サイズ(非圧縮時イメージサイズに対する倍数)
TileWidth	横方向のタイルサイズ(ピクセル単位)
TileHeight	縦方向のタイルサイズ(ピクセル単位)
NumrResLevel	解像度レベル(0～)
PrecinctWidth	横方向のプレシンクトサイズ(2 のべき乗"1～15"を指定、2^1～2^15)
PrecinctHeight	縦方向のプレシンクトサイズ(2 のべき乗"1～15"を指定、2^1～2^15)
CodeBlockWidth	横方向のコードブロックサイズ(2 のべき乗"2～10"を指定、2^2～2^10)
CodeBlockHeight	縦方向のコードブロックサイズ(2 のべき乗"2～10"を指定、2^2～2^10)
Comment	保存するコメント文字列(MAX:1023 バイト) 次のように設定してください。「Title + 0x0d + Subject + ・・・ + 0x0d」 Title: タイトル Subject: サブタイトル Author: 作成者 Comment: コメント ※16 進表記のため、Delphi は 0x を$に置き換えてください。
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対応イメージは 8 ビットグレースケールと 24 ビットカラーです。
一般的に JP2kFormat が False(0)であれば拡張子は"jpc"や"j2k"、True(0 以外)であれば"jp2"となります。
Reversible が False(0)の場合は、非可逆圧縮で ICT(非可逆コンポーネント変換), 9/7 非可逆ウェーブレット変換を行い、True(0 以外)の場合は可逆圧縮で RCT(可逆コンポーネント変換), 5/3 可逆ウェーブレット変換を行います。
Size が 0.01 であれば非圧縮時に比べてファイルサイズが 1/100 となります。Size が有効になるのは Reversible が False(0)の場合です。Size のデフォルト値は 0(何もしない)です。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合
IKJ2kFileSave("c:¥¥abc.jp2",・・・・・);
(2)FTP サーバーにファイルを保存する場合
IKJ2kFileSave("ftp://www.newtone.co.jp/image/abc.jp2;;;true;user;password",・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無("true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。
(3)HTTP(S)サーバーにファイルを保存する場合
IKJ2kFileSave("http://www.newtone.co.jp/image/abc.jp2;;;;user;password",・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略すると HTTP サーバからの処理となります。

【ImageKit7 との違い】
FTP サーバーや HTTP(S)サーバーに直接ファイルを保存することができるようになりました。

IKJ2kFileSaveMem

【機能】

ラスタイメージを JPEG2000 形式で Raw データに保存します。

【関数書式】

(1)C++Builder

BOOL IKJ2kFileSaveMem(HANDLE InHandle, HANDLE *OutHandle, BOOL JP2kFormat, BOOL Reversible, double Size, long TileWidth, long TileHeight, short NumrResLevel, short PrecinctWidth, short PrecinctHeight, short CodeBlockWidth, short CodeBlockHeight, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

(2)Delphi

function IKJ2kFileSaveMem(InHandle: THandle; var OutHandle: THandle; JP2kFormat, Reversible: LongBool; Size: Double; TileWidth, TileHeight: Longint; NumrResLevel, PrecinctWidth, PrecinctHeight, CodeBlockWidth, CodeBlockHeight: Smallint; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	ラスタイメージのメモリハンドル
OutHandle	保存する Raw データ
JP2kFormat	フォーマット(False(0):JPEG2000 Part1、True(0 以外):JPEG2000 Code Stream)
Reversible	圧縮方法(False(0):非可逆、True(0 以外):可逆)
Size	ファイルの保存サイズ(非圧縮時イメージサイズに対する倍数)
TileWidth	横方向のタイルサイズ(ピクセル単位)
TileHeight	縦方向のタイルサイズ(ピクセル単位)
NumrResLevel	解像度レベル(0～)
PrecinctWidth	横方向のプレシンクトサイズ(2 のべき乗"1～15"を指定、2^1～2^15)
PrecinctHeight	縦方向のプレシンクトサイズ(2 のべき乗"1～15"を指定、2^1～2^15)
CodeBlockWidth	横方向のコードブロックサイズ(2 のべき乗"2～10"を指定、2^2～2^10)
CodeBlockHeight	縦方向のコードブロックサイズ(2 のべき乗"2～10"を指定、2^2～2^10)
Comment	保存するコメント文字列(MAX:1023 バイト) 次のように設定してください。「Title + 0x0d + Subject + ・・・ + 0x0d」 Title: タイトル Subject: サブタイトル Author: 作成者 Comment:コメント ※16 進表記のため、Delphi は 0x を$に置き換えてください。
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】

実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】

対応イメージは 8 ビットグレースケールと 24 ビットカラーです。一般的に JP2kFormat が False(0)であれば拡張子は"jpc"や"j2k"、True(0 以外)であれば"jp2"となります。Reversible が False(0)の場合は、非可逆圧縮で ICT(非可逆コンポーネント変換), 9/7 非可逆ウェーブレット変換を行い、True(0 以外)の場合は可逆圧縮で RCT(可逆コンポーネント変換), 5/3 可逆ウェーブレット変換を行います。Size が 0.01 であれば非圧縮時に比べてファイルサイズが 1/100 となります。Size が有効になるのは Reversible が False(0)の場合です。Size のデフォルト値は 0(何もしない)です。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKJ2kFileSave** 関数と同じです。

IKJpegExifInfo

【機能】
Exif ファイルから主画像とサムネイル画像および Exif の各種情報を取得します。

【関数書式】
(1)C++Builder
BOOL IKJpegExifInfo(LPCTSTR FileName, PTR_EXIF_INFO ExifInfo);
(2)Delphi
function IKJpegExifInfo(FileName: PChar; var ExifInfo: EXIF_INFO): LongBool;

【引数】

名称	内容
FileName	情報を取得するファイル名
ExifInfo	取得する Exif 情報の構造体(ユーザ定義型)変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
成功した場合、ExifInfo のメンバー変数にそれぞれ値が設定されます。
サムネイル画像のメモリハンドルは ThumbImageInfo.ThumbImageHandle に設定されますので、メモリハンドルが不要になった段階で IKFreeMemory でメモリを解放してください。
EXIF_INFO については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」の EXIF_INFO のメンバー変数の説明**をご覧ください。
Exif(JPEG)の主画像を読み込む場合は **IKFileLoad** 関数や **IKJpegFileLoad** 関数をご使用ください。

【情報の取得について】
IKJpegExifInfo 関数の実行により、すべての項目が取得されるわけではなく、機器や画像の特性により設定されない項目もあります。
詳しくは、「社団法人　日本電子工業振興協会」発行の「JEIDA 規格　デジタルスチルカメラ用画像ファイルフォーマット規格(Exif)」を参照してください。

【画像情報の構成】
画像情報
　|
　├──── 主画像情報
　|
　├──── Exif 情報
　|
　├──── GPS 情報
　|
　└──── サムネイル画像情報

【ImageKit6 との違い】
EXIF_INFO 構造体にメンバー変数が追加されました。

【ImageKit7 との違い】
EXIF_INFO 構造体に IntOpeInfo が追加されました。

IKJpegExifInfoMem

【機能】
Exif 形式の Raw データから主画像とサムネイル画像および Exif の各種情報を取得します。

【関数書式】
(1)C++Builder
BOOL IKJpegExifInfoMem(HANDLE Handle, PTR_EXIF_INFO ExifInfo);
(2)Delphi
function IKJpegExifInfoMem(Handle: THandle; var ExifInfo: EXIF_INFO): LongBool;

【引数】

名称	内容
Handle	EXIF 形式の Raw データ
ExifInfo	取得する Exif 情報の構造体(ユーザ定義型)変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
成功した場合、ExifInfo のメンバー変数にそれぞれ値が設定されます。
サムネイル画像のメモリハンドルは ThumbImageInfo.ThumbImageHandle に設定されますので、メモリハンドルが不要になった段階で IKFreeMemory でメモリを解放してください。
EXIF_INFO については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」の EXIF_INFO のメンバー変数の説明**をご覧ください。
Exif(JPEG)の主画像を読み込む場合は **IKFileLoadMem** 関数や **IKJpegFileLoadMem** 関数をご使用ください。
ファイルあるいは Raw データから情報を取得する違いはありますが、動作としては **IKJpegExifInfo** 関数と同じです。

【情報の取得について】
IKJpegExifInfoMem 関数の実行により、すべての項目が取得されるわけではなく、機器や画像の特性により設定されない項目もあります。
詳しくは、「社団法人　日本電子工業振興協会」発行の「JEIDA 規格　デジタルスチルカメラ用画像ファイルフォーマット規格(Exif)」を参照してください。

【画像情報の構成】
画像情報
　|
　├──── 主画像情報
　|
　├──── Exif 情報
　|
　├──── GPS 情報
　|
　└──── サムネイル画像情報

【ImageKit7 との違い】
EXIF_INFO 構造体に IntOpeInfo が追加されました。

IKJpegFileLoad

【機能】
JPEG 形式のファイルからラスタイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKJpegFileLoad(LPCTSTR FileName, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKJpegFileLoad(FileName: PChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
対応イメージは 8 ビットグレースケール、24 ビットカラーです。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

IKJpegFileLoadMem

【機能】
JPEG 形式の Raw データからラスタイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKJpegFileLoadMem(HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKJpegFileLoadMem(Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	JPEG 形式の Raw データ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
対応イメージは 8 ビットグレースケール、24 ビットカラーです。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKJpegFileLoad** 関数と同じです。

IKJpegFileSave,IKJpegFileSaveEx

【機能】

ラスタイメージを JPEG 形式でファイルに保存します。**IKJpegFileSaveEx** では Exif 情報も併せて保存します。

【関数書式】

(1)C++Builder

BOOL **IKJpegFileSave**(LPCTSTR FileName, HANDLE Handle, int Type, int Quality, int SubSamp, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

BOOL **IKJpegFileSaveEx**(LPCTSTR FileName, HANDLE Handle, int Type, int Quality, int SubSamp, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button, PTR_EXIF_INFO ExifInfo);

(2)Delphi

function **IKJpegFileSave**(FileName: PChar; Handle: THandle; Type_, Quality, SubSamp: Integer; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

function **IKJpegFileSaveEx**(FileName: PChar; Handle: THandle; Type_, Quality, SubSamp: Integer; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar; var ExifInfo: EXIF_INFO): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ラスタイメージのメモリハンドル
Type	0:基本 DCT、1:プログレッシブ DCT
Quality	JPEG 圧縮品質係数(0～100/推奨 75)
SubSamp	サブサンプリング(24 ビットカラー時有効)、Y:Cb:Cr(0: 4:1:1, 1: 4:2:2, 2: 4:4:4)
Comment	保存するコメント文字列(MAX:1023 バイト)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列
ExifInfo	Exif 情報の構造体(ユーザ定義型)変数、**IKJpegFileSaveEx** で使用

【戻り値】

実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】

対応イメージは 8 ビットグレースケールと 24 ビットカラーです。SubSamp の Y は輝度情報、Cr と Cb は色相情報を表します。ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
EXIF_INFO については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」の EXIF_INFO のメンバー変数の説明**をご覧ください。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合

IKJpegFileSave("c:¥¥abc.jpg",・・・・・);

(2)FTP サーバーにファイルを保存する場合

IKJpegFileSave("ftp://www.newtone.co.jp/image/abc.jpg;;;true;user;password",・・・・・);

(*)FileName を設定する際の順番

FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。

（文字列をセミコロンで区切る）
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無（"true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。

(3)HTTP(S)サーバーにファイルを保存する場合
IKJpegFileSave("http://www.newtone.co.jp/image/abc.jpg;;;;user;password",・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。（文字列をセミコロンで区切る）
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略すると HTTP サーバからの処理となります。

【ImageKit7 との違い】
・FTP サーバーや HTTP(S)サーバーに直接ファイルを保存することができるようになりました。
・IKJpegFileSaveEx が追加されました。

IKJpegFileSaveMem,IKJpegFileSaveExMem

【機能】

ラスタイメージを JPEG 形式で Raw データに保存します。**IKJpegFileSaveExMem** では Exif 情報も併せて保存します。

【関数書式】

(1)C++Builder

BOOL **IKJpegFileSaveMem**(HANDLE InHandle, HANDLE *OutHandle, int Type, int Quality, int SubSamp, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
BOOL **IKJpegFileSaveExMem**(HANDLE InHandle, HANDLE *OutHandle, int Type, int Quality, int SubSamp, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button, PTR_EXIF_INFO ExifInfo);

(2)Delphi

function **IKJpegFileSaveMem**(InHandle: THandle; var OutHandle: THandle; Type_, Quality, SubSamp: Integer; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;
function **IKJpegFileSaveExMem**(InHandle: THandle; var OutHandle: THandle; Type_, Quality, SubSamp: Integer; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar; var ExifInfo: EXIF_INFO): LongBool;

【引数】

名称	内容
InHandle	ラスタイメージのメモリハンドル
OutHandle	保存する Raw データ
Type	0:基本 DCT、1:プログレッシブ DCT
Quality	JPEG 圧縮品質係数(0～100/推奨 75)
SubSamp	サブサンプリング(24 ビットカラー時有効)、Y:Cb:Cr(0: 4:1:1, 1: 4:2:2, 2: 4:4:4)
Comment	保存するコメント文字列(MAX:1023 バイト)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列
ExifInfo	Exif 情報の構造体(ユーザ定義型)変数、**IKJpegFileSaveExMem** で使用

【戻り値】

実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】

対応イメージは 8 ビットグレースケールと 24 ビットカラーです。SubSamp の Y は輝度情報、Cr と Cb は色相情報を表します。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKJpegFileSave(Ex)**関数と同じです。

【ImageKit7 との違い】

IKJpegFileSaveExMem が追加されました。

IKOpenFileDialog

【機能】
プレビューおよびファイル情報付きのファイルオープンダイアログを表示します。

【関数書式】
(1)C++Builder
BOOL IKOpenFileDialog(HWND hWnd, LPTSTR FileName, LPCTSTR FilePath, LPCTSTR FileExt, BOOL ExtendedDialog, BOOL Preview, BOOL FileInfo);
(2)Delphi
function IKOpenFileDialog(hWnd: HWND; FileName, FilePath, FileExt: PChar; ExtendedDialog, Preview, FileInfo: LongBool): LongBool;

【引数】

名称	内容
hWnd	ウィンドウハンドル
FileName	選択されたファイル名
FilePath	初期表示のフォルダの名称 ヌル文字列を設定した場合はカレントフォルダを参照します。
FileExt	初期表示のファイルの拡張子(アルファベットの大文字と小文字は区別しない) 複数の種類を設定する場合は、"BMP;JPG"のように";"で区切ります。 "*"を設定すると、種類のところに「全てのファイル」は表示されません。 拡張子を"<JPG>"のように<>で囲むとデフォルトとして表示できます。
ExtendedDialog	False(0):Preview と FileInfo のチェックボックスを非表示 True(0 以外):Preview と FileInfo のチェックボックスを表示
Preview	初期表示時のプレビューの設定 False(0):プレビュー表示なし、True(0 以外):プレビュー表示あり
FileInfo	初期表示時のファイル情報の設定 False(0):ファイル情報表示なし、True(0 以外):ファイル情報表示あり

【戻り値】
キャンセルを選択した場合は False(0)、「開く」を選択した場合は True(0 以外)が返されます。

【解説】
FilePath と FileExt で設定されたファイルをダイアログのリストに初期表示します。
「開く」ボタンを選択するとファイル名がフルパスで FileName に設定されます。
ExtendedDialog が True(0 以外)の場合は、ダイアログ初期表示時にプレビューとファイル情報表示のチェックボックスが表示されます。そのため、Preview と FileInfo が False(0)でもダイアログ表示時にそれぞれ表示することができます。
プレビューでベクトルイメージを参照する場合は、事前に
Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll の IKVectorGdipStart 関数を実行する必要があります。

IKOpenFileDlg

【機能】
プレビュー付きのファイルオープンダイアログを表示します。

【関数書式】
(1)C++Builder
BOOL IKOpenFileDlg(HWND hWnd, LPTSTR FileName, LPCTSTR FilePath, LPCTSTR FileExt, BOOL Preview);
(2)Delphi
function IKOpenFileDlg(hWnd: HWND; FileName, FilePath, FileExt: PChar; Preview: LongBool): LongBool;

【引数】

名称	内容
hWnd	ウィンドウハンドル
FileName	選択されたファイル名
FilePath	初期表示のフォルダの名称 ヌル文字列を設定した場合はカレントフォルダを参照します。
FileExt	初期表示のファイルの拡張子(アルファベットの大文字と小文字は区別しない) 複数の種類を設定する場合は、"BMP;JPG"のように";"で区切ります。 "*"を設定すると、種類のところに「全てのファイル」は表示されません。 拡張子を"<JPG>"のように<>で囲むとデフォルトとして表示できます。
Preview	初期表示時のプレビューの設定(ダイアログ表示時にも設定可) False(0):プレビュー表示なし、True(0 以外):プレビュー表示あり

【戻り値】
キャンセルを選択した場合は False(0)、「開く」を選択した場合は True(0 以外)が返されます。

【解説】
FilePath と FileExt で設定されたファイルをダイアログのリストに初期表示します。
「開く」ボタンを選択するとファイル名がフルパスで FileName に設定されます。
プレビューでベクトルイメージを参照する場合は、事前に
Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll の IKVectorGdipStart 関数を実行する必要があります。

IKPcxFileLoad

【機能】
PCX 形式のファイルからラスタイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKPcxFileLoad(LPCTSTR FileName, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKPcxFileLoad(FileName: PChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
対応イメージは 1,4,8,24 ビットカラーです。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

IKPcxFileLoadMem

【機能】

PCX 形式の Raw データからラスタイメージをメモリ上に読み込みます。

【関数書式】

(1)C++Builder

HANDLE IKPcxFileLoadMem(HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

(2)Delphi

function IKPcxFileLoadMem(Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	PCX 形式の Raw データ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】

ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】

対応イメージは 1,4,8,24 ビットカラーです。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKPcxFileLoad** 関数と同じです。

IKPcxFileSave

【機能】
ラスタイメージを PCX 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKPcxFileSave(LPCTSTR FileName, HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKPcxFileSave(FileName: PChar; Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ラスタイメージのメモリハンドル
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対応イメージは 1,4,8,24 ビットカラーです。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合
IKPcxFileSave("c:¥¥abc.pcx",・・・・・);
(2)FTP サーバーにファイルを保存する場合
IKPcxFileSave("ftp://www.newtone.co.jp/image/abc.pcx;;;true;user;password",・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無("true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。
(3)HTTP(S)サーバーにファイルを保存する場合
IKPcxFileSave("http://www.newtone.co.jp/image/abc.pcx;;;;user;password",・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。(文字列をセミコロンで区切る)

なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略すると HTTP サーバからの処理となります。

【ImageKit7 との違い】
FTP サーバーや HTTP(S)サーバーに直接ファイルを保存することができるようになりました。

IKPcxFileSaveMem

【機能】
ラスタイメージを PCX 形式で Raw データに保存します。

【関数書式】
(1)C++Builder
BOOL IKPcxFileSaveMem(HANDLE InHandle, HANDLE *OutHandle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKPcxFileSaveMem(InHandle: THandle; var OutHandle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	ラスタイメージのメモリハンドル
OutHandle	保存する Raw データ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対応イメージは 1,4,8,24 ビットカラーです。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKPcxFileSave** 関数と同じです。

IKPDFAddImage

【機能】
PDF の作成で画像を追加します。

【関数書式】
(1)C++Builder
BOOL IKPDFAddImage(PTR_SAVE_PDF_INFO PDFSetInfo, HANDLE ImageHandle, unsigned int x, unsigned int y, unsigned int size);
(2)Delphi
function IKPDFAddImage(var pdf_set: SAVE_PDF_INFO; ImageHandle: THandle; x: UINT; y: UINT; size: UINT): LongBool;

【引数】

名称	内容
PDFSetInfo	PDF 作成の条件を設定する構造体(ユーザ定義型)変数
ImageHandle	対象となる画像の Raw データのハンドル(JPEG と PNG 形式が対象)
x	画像を追加する横方向の位置(左下を原点に mm 単位で指定)
y	画像を追加する縦方向の位置(左下を原点に mm 単位で指定)
size	画像を追加するサイズの指定(mm 単位)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PDFSetInfo は IKPDFStart、IKPDFAddPage、IKPDFEnd 関数でも使用します。
SAVE_PDF_INFO については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」の SAVE_PDF_INFO のメンバー変数の説明**をご覧ください。
画像データの縦サイズと横サイズのうち大きいサイズを size に合わせて指定した x,y の位置に画像を設定します。
画像データの縦横比は保持したまま、x,y,size は mm 単位での指定です。
当関数実行後に画像や位置・サイズを変更するなどして当関数を繰り返せば、一つのページ内に複数の画像を追加することも可能です。

IKPDFAddPage

【機能】
PDF の作成でページを追加します。

【関数書式】
(1)C++Builder
BOOL IKPDFAddPage(PTR_SAVE_PDF_INFO PDFSetInfo, LPCSTR DocumentSize, BOOL Landscape, unsigned int DocumentWidth, unsigned int DocumentHeight);
(2)Delphi
function IKPDFAddPage(var pdf_set: SAVE_PDF_INFO; DocumentSize: LPCSTR; Landscape: LongBool; DocumentWidth: UINT; DocumentHeight: UINT): LongBool;

【引数】

名称	内容
PDFSetInfo	PDF 作成の条件を設定する構造体(ユーザ定義型)変数
DocumentSize	文書のサイズ(A0～A9、B0～B9、LETTER、LEGAL)
Landscape	文書の向きの指定(TRUE ランドスケープ、FALSE ポートレート)
DocumentWidth	文書の幅のサイズの指定(mm 単位)
DocumentHeight	文書の高さのサイズの指定(mm 単位)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PDFSetInfo は IKPDFStart、IKPDFAddImage、IKPDFEnd 関数でも使用します。
SAVE_PDF_INFO については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」の SAVE_PDF_INFO のメンバー変数の説明**をご覧ください。
DocumentSize に A4 などのサイズを渡した場合、DocumentWidth と DocumentHeight の値は無効になります。
DocumentWidth と DocumentHeight の値を有効にする場合は DocumentSize に NULL,nil,空文字列を渡してください。

DocumentSize に指定可能なサイズ:
A0、A1、A2、A3、A4、A5、A6、A7、A8、A9、B0、B1、B2、B3、B4、B5、B6、B7、B8、B9、LETTER、LEGAL
※大文字、小文字は問いません。

IKPDFEnd

【機能】
PDF の作成処理を終了します。

【関数書式】
(1)C++Builder
BOOL IKPDFEnd(PTR_SAVE_PDF_INFO PDFSetInfo, LPCTSTR PDFFileName);
(2)Delphi
function IKPDFEnd(var pdf_set: SAVE_PDF_INFO; PDFFileName: LPCTSTR): LongBool;

【引数】

名称	内容
PDFSetInfo	PDF 作成の条件を設定する構造体(ユーザ定義型)変数
PDFFileName	保存するファイル名

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PDFSetInfo は IKPDFStart、IKPDFAddPage、IKPDFAddImage 関数でも使用します。
SAVE_PDF_INFO については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」の SAVE_PDF_INFO のメンバー変数の説明**をご覧ください。

IKPDFStart

【機能】
PDF の作成処理を開始します。

【関数書式】
(1)C++Builder
BOOL IKPDFStart(PTR_SAVE_PDF_INFO PDFSetInfo);
(2)Delphi
function IKPDFStart(var pdf_set: SAVE_PDF_INFO): LongBool;

【引数】

名称	内容
PDFSetInfo	PDF 作成の条件を設定する構造体(ユーザ定義型)変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PDFSetInfo を初期化したのち、所定のメンバーに適切な値を設定して当関数に渡してください。PDFSetInfo は IKPDFAddPage、IKPDFAddImage、IKPDFEnd 関数でも使用します。
SAVE_PDF_INFO については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」の SAVE_PDF_INFO のメンバー変数の説明**をご覧ください。

コード例:
(1)C++Builder

```
HANDLE ImageHandle
SAVE_PDF_INFO PDFSetInfo;

ImageHandle = IKFileLoadAsRawData("C:¥¥PNG¥¥load.png", 0, "", "", "");
if (ImageHandle == 0) return;

memset(&PDFSetInfo, 0, sizeof(SAVE_PDF_INFO));
lstrcpy(PDFSetInfo.OwnerPassword, "abcd");
lstrcpy(PDFSetInfo.Application, "PDF 作成ツール");
lstrcpy(PDFSetInfo.Author, "Newtone Corp.");
PDFSetInfo.EnablePrint = TRUE;

if (IKPDFStart(&PDFSetInfo) == FALSE)
{
  IKFreeMemory(ImageHandle);
  return;
}
IKPDFAddPage(&PDFSetInfo, "A4", FALSE, 0, 0);
IKPDFAddImage(&PDFSetInfo, ImageHandle, 30, 50, 100);
IKPDFEnd(&PDFSetInfo, "C:¥¥PDF¥¥save.pdf");

IKFreeMemory(ImageHandle);
```

(2)Delphi

```
var
  ImageHandle: THandle;
  PDFSetInfo: SAVE_PDF_INFO;
begin
  ImageHandle := IKFileLoadAsRawData('C:¥PNG¥load.png', 0, '', '', '');
  if (ImageHandle = 0) then Exit;
```

```
  FillChar(PDFSetInfo, SizeOf(PDFSetInfo), 0);
  StrPCopy(PDFSetInfo.OwnerPassword, 'abcd');
  StrPCopy(PDFSetInfo.Application, 'PDF 作成ツール');
  StrPCopy(PDFSetInfo.Author, 'Newtone Corp.');
  PDFSetInfo.EnablePrint := True;

  if (IKPDFStart(PDFSetInfo) = False) then
  begin
    IKFreeMemory(ImageHandle);
    Exit;
  end;
  IKPDFAddPage(PDFSetInfo, 'A4', False, 0, 0);
  IKPDFAddImage(PDFSetInfo, ImageHandle, 30, 50, 100);
  IKPDFEnd(PDFSetInfo, 'C:¥PDF¥save.pdf');

  IKFreeMemory(ImageHandle);
end;
```

IKPngFileLoad,IKPngFileLoadEx

【機能】
PNG 形式のファイルからラスタイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE **IKPngFileLoad**(LPCTSTR FileName, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
HANDLE **IKPngFileLoadEx**(LPCTSTR FileName, int Alpha, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function **IKPngFileLoad**(FileName: PChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;
function **IKPngFileLoadEx**(FileName: PChar; Alpha: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
Alpha	アルファチャンネル(IKPngFileLoadEx で使用) 0: RGB プレーンのみ読み込み 1: A プレーンを考慮して RGB プレーンを補正 2: RGBA プレーンをそのまま読み込む(32 ビットイメージとして出力)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
対応イメージは 1,4,8,24 ビットカラーです。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

IKPngFileLoadMem,IKPngFileLoadExMem

【機能】

PNG 形式の Raw データからラスタイメージをメモリ上に読み込みます。

【関数書式】

(1)C++Builder

HANDLE **IKPngFileLoadMem**(HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

HANDLE **IKPngFileLoadExMem**(HANDLE Handle, int Alpha, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

(2)Delphi

function **IKPngFileLoadMem**(Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

function **IKPngFileLoadExMem**(Handle: THandle; Alpha: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	PNG 形式の Raw データ
Alpha	アルファチャンネル(IKPngFileLoadExMem で使用) 0: RGB プレーンのみ読み込み 1: A プレーンを考慮して RGB プレーンを補正 2: RGBA プレーンをそのまま読み込む(32 ビットイメージとして出力)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】

ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】

対応イメージは 1,4,8,24 ビットカラーです。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKPngFileLoad(Ex)**関数と同じです。

IKPngFileSave,IKPngFileSaveEx

【機能】
ラスタイメージを PNG 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL **IKPngFileSave**(LPCTSTR FileName, HANDLE Handle, BOOL Interlace, BOOL Trans, BYTE Red, BYTE Green, BYTE Blue, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
BOOL **IKPngFileSaveEx**(LPCTSTR FileName, HANDLE Handle, BOOL Interlace, BOOL Trans, BYTE Red, BYTE Green, BYTE Blue, BOOL Alpha, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function **IKPngFileSave**(FileName: PChar; Handle: THandle; Interlace, Trans: LongBool; Red, Green, Blue: Byte; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;
function **IKPngFileSaveEx**(FileName: PChar; Handle: THandle; Interlace, Trans: LongBool; Red, Green, Blue: Byte; Alpha: LongBool; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ラスタイメージのメモリハンドル
Interlace	インタレース(False(0):設定なし、True(0 以外):設定あり)
Trans	透過(False(0):設定なし、True(0 以外):設定あり)
Red	透過パレットの赤 (0～255) ※Trans が True(0 以外)の場合に有効となります。
Green	透過パレットの緑 (0～255) ※Trans が True(0 以外)の場合に有効となります。
Blue	透過パレットの青 (0～255) ※Trans が True(0 以外)の場合に有効となります。
Alpha	アルファチャンネル(IKPngFileSaveEx で使用) False(0): RGB として保存、IKPngFileSave と同じ動作 True(0 以外): RGBA として保存
Comment	保存するコメント文字列(MAX:1023 バイト) 次のように設定してください。「Title + 0x0d + Aother + ・・・ + 0x0d」 Title: タイトル Aothor: 作成者 Description: 説明 Copyright: 版権 Creation Time: 時間 Software: ソフトウエア Disclaimer: 権利放棄 Warning: 警告 Source: 入力機器 Comment: コメント ※16 進表記のため、Delphi は 0x を$に置き換えてください。
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対応イメージは 1,4,8,24 ビットカラーです。

IKPngFileSaveEx 関数で Alpha を True(0 以外)に設定する場合は、Handle に RGBA の 32 ビットイメージのメモリハンドルを設定する必要があります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合
IKPngFileSave(″c:¥¥abc.png″,・・・・・);
(2)FTP サーバーにファイルを保存する場合
IKPngFileSave(″ftp://www.newtone.co.jp/image/abc.png;;;true;user;password″,・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無(″true″または″false″ - 大小文字関係なし)を省略するとパッシブモード接続となります。
(3)HTTP(S)サーバーにファイルを保存する場合
IKPngFileSave(″http://www.newtone.co.jp/image/abc.png;;;;user;password″,・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、″HTTPS″を省略すると HTTP サーバからの処理となります。

【参照】
「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」の IKMakeRGBAImage 関数

【ImageKit7 との違い】
FTP サーバーや HTTP(S)サーバーに直接ファイルを保存することができるようになりました。

IKPngFileSaveMem,IKPngFileSaveExMem

【機能】

ラスタイメージを PNG 形式で Raw データに保存します。

【関数書式】

(1)C++Builder

BOOL **IKPngFileSaveMem**(HANDLE InHandle, HANDLE *OutHandle, BOOL Interlace, BOOL Trans, BYTE Red, BYTE Green, BYTE Blue, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

BOOL **IKPngFileSaveExMem**(HANDLE InHandle, HANDLE *OutHandle, BOOL Interlace, BOOL Trans, BYTE Red, BYTE Green, BYTE Blue, BOOL Alpha, LPCSTR Comment, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

(2)Delphi

function **IKPngFileSaveMem**(InHandle: THandle; var OutHandle: THandle; Interlace, Trans: LongBool; Red, Green, Blue: Byte; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

function **IKPngFileSaveExMem**(InHandle: THandle; var OutHandle: THandle; Interlace, Trans: LongBool; Red, Green, Blue: Byte; Alpha: LongBool; Comment: PAnsiChar; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	ラスタイメージのメモリハンドル
OutHandle	保存する Raw データ
Interlace	インタレース(False(0):設定なし、True(0 以外):設定あり)
Trans	透過(False(0):設定なし、True(0 以外):設定あり)
Red	透過パレットの赤 (0～255) ※Trans が True(0 以外)の場合に有効となります。
Green	透過パレットの緑 (0～255) ※Trans が True(0 以外)の場合に有効となります。
Blue	透過パレットの青 (0～255) ※Trans が True(0 以外)の場合に有効となります。
Alpha	アルファチャンネル(IKPngFileSaveExMem で使用) False(0): RGB として保存、IKPngFileSaveMem と同じ動作 True(0 以外): RGBA として保存
Comment	保存するコメント文字列(MAX:1023 バイト) 次のように設定してください。「Title + 0x0d + Aother + ・・・ + 0x0d」 Title: タイトル Aothor: 作成者 Description: 説明 Copyright: 版権 Creation Time: 時間 Software: ソフトウエア Disclaimer: 権利放棄 Warning: 警告 Source: 入力機器 Comment: コメント ※16 進表記のため、Delphi は 0x を$に置き換えてください。
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】

実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】

対応イメージは 1,4,8,24 ビットカラーです。
IKPngFileSaveExMem 関数で Alpha を True(0 以外)に設定する場合は、Handle に RGBA の 32 ビットイメージのメモリハンドルを設定する必要があります。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKPngFileSave(Ex)**関数と同じです。

【参照】
「Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll」の IKMakeRGBAImage 関数

IKRGBBmpPlaneFileLoad

【機能】
RGB プレーン毎に保存してある BMP ファイルからラスタイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKRGBBmpPlaneFileLoad(LPCTSTR RedFileName, LPCTSTR GreenFileName, LPCTSTR BlueFileName);
(2)Delphi
function IKRGBBmpPlaneFileLoad(RedFileName, GreenFileName, BlueFileName: PChar): THandle;

【引数】

名称	内容
RedFileName	R プレーンとして読み込むファイル名
GreenFileName	G プレーンとして読み込むファイル名
BlueFileName	B プレーンとして読み込むファイル名

【戻り値】
ラスタイメージのメモリハンドル（実行に失敗した場合は 0 が返されます）。

【解説】
RGB それぞれのプレーンを表すイメージは 8 ビットグレーで、出力イメージは 24 ビットカラーです。

IKRGBBmpPlaneFileSave

【機能】
ラスタイメージを RGB プレーン毎に BMP 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKRGBBmpPlaneFileSave(LPCTSTR RedFileName, LPCTSTR GreenFileName, LPCTSTR BlueFileName, HANDLE Handle);
(2)Delphi
function IKRGBBmpPlaneFileSave(RedFileName, GreenFileName, BlueFileName: PChar; Handle: THandle): LongBool;

【引数】

名称	内容
RedFileName	R プレーンとして保存するファイル名
GreenFileName	G プレーンとして保存するファイル名
BlueFileName	B プレーンとして保存するファイル名
Handle	ラスタイメージのメモリハンドル

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
保存対象イメージは 24 ビットカラーです。Handle に設定されているメモリハンドルを RGB のそれぞれのプレーン毎にファイルに保存します。保存されるイメージは 8 ビットグレーとなります。

IKSaveFileDialog

【機能】
プレビューおよびファイル情報付きのファイルセーブダイアログを表示します。

【関数書式】
(1)C++Builder
BOOL IKSaveFileDialog(HWND hWnd, LPTSTR FileName, LPCTSTR FilePath, LPCTSTR FileExt, BOOL ExtendedDialog, BOOL Preview, LPINT FileType, BOOL FileInfo);
(2)Delphi
function IKSaveFileDialog(hWnd: HWND; FileName, FilePath, FileExt: PChar; ExtendedDialog, Preview: LongBool; var FileType: Integer; FileInfo: LongBool): LongBool;

【引数】

名称	内容
hWnd	ウィンドウハンドル
FileName	選択されたファイル名もしくは初期表示のファイル名
FilePath	初期表示のフォルダの名称(ヌル文字列を設定した場合は、カレントフォルダを参照)
FileExt	初期表示のファイルの拡張子(アルファベットの大文字と小文字は区別しない) 複数の種類を設定する場合は、"BMP;JPG"のように";"で区切ります。 "*"を設定すると、種類のところに「全てのファイル」は表示されません。 拡張子を"<JPG>"のように<>で囲むとデフォルトとして表示できます。
ExtendedDialog	False(0):Preview と FileInfo のチェックボックスを非表示 True(0 以外):Preview と FileInfo のチェックボックスを表示
Preview	初期表示時のプレビューの設定 False(0):プレビュー表示なし、True(0 以外):プレビュー表示あり
FileType	選択されたファイルの種類 1:BMP,2:JPEG,3:GIF,4:TIFF,5:PNG,6:FPX,7:PCX,8:WMF,9:EMF,10:DXF,11:SVG 12:JPEG2000,13:JPEG2000(Code Stream),14:SXF(p21),15:SXF(sfc)
FileInfo	初期表示時のファイル情報の設定 False(0):ファイル情報表示なし、True(0 以外):ファイル情報表示あり

【戻り値】
キャンセルを選択した場合は False(0)、保存を選択した場合は True(0 以外)が返されます。

【解説】
FilePath と FileExt の条件に合ったファイルをダイアログのリストに初期表示します。ただし、FileName にフルパスが設定されている場合は FilePath は無効です。
保存ボタンを選択するとファイル名がフルパスで FileName に、FileType に選択されたファイルの種類が設定されます。
ExtendedDialog が True(0 以外)の場合は、Preview と FileInfo が False(0)でもダイアログ表示時にそれぞれ表示することができます。
プレビューでベクトルイメージを参照する場合は、事前に
Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll の IKVectorGdipStart 関数を実行する必要があります。

例:
(1)C++Builder

```
TCHAR FileName[256];
BOOL Ret;
int FileType;

memset(FileName, 0, sizeof(FileName));          //この場合は、FilePath 有効
```

```
//FileName にフルパスでファイル名を設定すると FilePath 無効
//lstrcpy(FileName, "C:¥¥Images¥¥001.jpg");

Ret = IKSaveFileDialog(Form1->Handle, FileName, "C:¥¥My Pictures", "*;BMP;JPG;", FALSE, FALSE, &FileType,
FALSE,);
```

(2)Delphi

```
FileName: array [0..255] of Char;
Ret: LongBool;
FileType: Integer;

FillChar(FileName, SizeOf(FileName), 0);          //この場合は、FilePath 有効
//FileName にフルパスでファイル名を設定すると FilePath 無効
//StrPCopy(FileName, 'C:¥Images¥001.jpg');
Ret = IKSaveFileDialog(Form1.Handle, FileName, 'C:¥My Pictures', '*;BMP;JPG;', False, False, FileType, False);
```

IKSaveFileDlg,IKSaveFileDlgEx

【機能】
プレビュー付きのファイルセーブダイアログを表示します。

【関数書式】
(1)C++Builder
BOOL **IKSaveFileDlg**(HWND hWnd, LPTSTR FileName, LPCTSTR FilePath, LPCTSTR FileExt, BOOL Preview);
BOOL **IKSaveFileDlgEx**(HWND hWnd, LPTSTR FileName, LPCTSTR FilePath, LPCTSTR FileExt, BOOL Preview, LPINT FileType);
(2)Delphi
function **IKSaveFileDlg**(hWnd: HWND; FileName, FilePath, FileExt: PChar; Preview: LongBool): LongBool;
function **IKSaveFileDlgEx**(hWnd: HWND; FileName, FilePath, FileExt: PChar; Preview: LongBool; var FileType: Integer): LongBool;

【引数】

名称	内容
hWnd	ウィンドウハンドル
FileName	選択されたファイル名もしくは初期表示のファイル名
FilePath	初期表示のフォルダの名称(ヌル文字列を設定した場合は、カレントフォルダを参照)
FileExt	初期表示のファイルの拡張子(アルファベットの大文字と小文字は区別しない) 複数の種類を設定する場合は、"BMP;JPG"のように";"で区切ります。 "*"を設定すると、種類のところに「全てのファイル」は表示されません。 拡張子を"<JPG>"のように<>で囲むとデフォルトとして表示できます。
Preview	初期表示時のプレビューの設定(ダイアログ表示時にも設定可) False(0):プレビュー表示なし、True(0 以外):プレビュー表示あり
FileType	選択されたファイルの種類(IKSaveFileDlgEx で使用) 1:BMP,2:JPEG,3:GIF,4:TIFF,5:PNG,6:FPX,7:PCX,8:WMF,9:EMF,10:DXF,11:SVG 12:JPEG2000,13:JPEG2000(Code Stream),14:SXF(p21),15:SXF(sfc)

【戻り値】
キャンセルを選択した場合は False(0)、保存を選択した場合は True(0 以外)が返されます。

【解説】
FilePath とFileExt の条件に合ったファイルをダイアログのリストに初期表示します。ただし、FileName にフルパスが設定されている場合は FilePath は無効です。
保存ボタンを選択するとファイル名がフルパスで FileName に設定されます。
IKSaveFileDlgEx では、保存ボタンを選択すると FileType に選択されたファイルの種類が設定されます。
プレビューでベクトルイメージを参照する場合は、事前に
Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll の **IKVectorGdipStart** 関数を実行する必要があります。

例:
(1)C++Builder

```
TCHAR FileName[256];
BOOL Ret;
int FileType;

memset(FileName, 0, sizeof(FileName));          //この場合は、FilePath 有効
//FileName にフルパスでファイル名を設定すると FilePath 無効
//lstrcpy(FileName, "C:¥¥Images¥¥001.jpg");

Ret = IKSaveFileDlgEx(Form1->Handle, FileName, "C:¥¥My Pictures", "*;BMP;JPG;", FALSE, &FileType);
```

(2)Delphi

```
FileName: array [0..255] of Char;
Ret: LongBool;
FileType: Integer;

FillChar(FileName, SizeOf(FileName), 0);          //この場合は、FilePath 有効
//FileName にフルパスでファイル名を設定すると FilePath 無効
//StrPCopy(FileName, 'C:¥Images¥001.jpg');
Ret = IKSaveFileDlgEx(Form1.Handle, FileName, 'C:¥My Pictures', '*;BMP;JPG;', False, FileType);
```

IKSvgFileLoad

【機能】
SVG 形式のファイルからベクトルイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKSvgFileLoad(LPCTSTR FileName, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKSvgFileLoad(FileName: PChar; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
Width	読み込むイメージの幅(ピクセル)
Height	読み込むイメージの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Width もしくは Height が 0 以下の場合は SVG ファイルに格納されているサイズで読み込みます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

(注意)
読み込まれるイメージの縦横のサイズは、設定した矩形内(Width * Height)で縦横比を考慮した最も大きなサイズとなりますので、設定したサイズ通りにならない場合があります。

IKSvgFileLoadMem

【機能】
SVG 形式の Raw データからベクトルイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKSvgFileLoadMem(HANDLE Handle, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKSvgFileLoadMem(Handle: THandle; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	SVG 形式の Raw データ
Width	読み込むイメージの幅(ピクセル)
Height	読み込むイメージの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Width もしくは Height が 0 以下の場合は SVG 形式の Raw データに格納されているサイズで読み込みます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKSvgFileLoad** 関数と同じです。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

(注意)
読み込まれるイメージの縦横のサイズは、設定した矩形内(Width * Height)で縦横比を考慮した最も大きなサイズとなりますので、設定したサイズ通りにならない場合があります。

IKSvgFileSave

【機能】
ベクトルイメージのメモリハンドルを SVG 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKSvgFileSave(LPCTSTR FileName, HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKSvgFileSave(FileName: PChar; Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ベクトルイメージのメモリハンドル
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合
IKSvgFileSave("c:¥¥abc.svg",・・・・・);
(2)FTP サーバーにファイルを保存する場合
IKSvgFileSave("ftp://www.newtone.co.jp/image/abc.svg;;;true;user;password",・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無("true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。
(3)HTTP(S)サーバーにファイルを保存する場合
IKSvgFileSave("http://www.newtone.co.jp/image/abc.svg;;;;user;password",・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。(文字列をセミコロンで

区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略すると HTTP サーバからの処理となります。

【ImageKit7 との違い】
FTP サーバーや HTTP(S)サーバーに直接ファイルを保存することができるようになりました。

IKSvgFileSaveMem

【機能】
ベクトルイメージのメモリハンドルを SVG 形式で Raw データに保存します。

【関数書式】
(1)C++Builder
BOOL IKSvgFileSaveMem(HANDLE InHandle, HANDLE *OutHandle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKSvgFileSaveMem(InHandle: THandle; var OutHandle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	ベクトルイメージのメモリハンドル
OutHandle	保存する Raw データ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKSvgFileSave** 関数と同じです。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

IKSxfP21FileLoad,IKSxfSfcFileLoad

【機能】
SXF 形式のファイルからベクトルイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE **IKSxfP21FileLoad**(LPCTSTR FileName, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
HANDLE **IKSxfSfcFileLoad**(LPCTSTR FileName, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function **IKSxfP21FileLoad**(FileName: PChar; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;
function **IKSxfSfcFileLoad**(FileName: PChar; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
Width	読み込むイメージの幅(ピクセル)
Height	読み込むイメージの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
p21 形式のファイルを読み込む場合は **IKSxfP21FileLoad** 関数を、sfc 形式のファイルを読み込む場合は **IKSxfSfcFileLoad** 関数を使用します。
Width もしくは Height が 0 以下の場合は幅 800 高さ 600 に設定します。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

(注意)
読み込まれるイメージの縦横のサイズは、設定した矩形内(Width * Height)で縦横比を考慮した最も大きなサイズとなりますので、設定したサイズ通りにならない場合があります。

IKSxfP21FileLoadMem,IKSxfSfcFileLoadMem

【機能】
SXF 形式の Raw データからベクトルイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE **IKSxfP21FileLoadMem**(HANDLE Handle, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
HANDLE **IKSxfSfcFileLoadMem**(HANDLE Handle, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function **IKSxfP21FileLoadMem**(Handle: THandle; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;
function **IKSxfSfcFileLoadMem**(Handle: THandle; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	SXF 形式の Raw データ
Width	読み込むイメージの幅(ピクセル)
Height	読み込むイメージの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
p21 形式の Raw データを読み込む場合は **IKSxfP21FileLoadMem** 関数を、sfc 形式の Raw データを読み込む場合は **IKSxfSfcFileLoadMem** 関数を使用します。
Width もしくは Height が 0 以下の場合は幅 800 高さ 600 に設定します。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKSxfP21FileLoad**,**IKSxfSfcFileLoad** 関数と同じです。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

(注意)
読み込まれるイメージの縦横のサイズは、設定した矩形内(Width * Height)で縦横比を考慮した最も大きなサイズとなりますので、設定したサイズ通りにならない場合があります。内部で一時的にテンポラリファイルを作成します。

IKSxfP21FileSave,IKSxfSfcFileSave

【機能】
ベクトルイメージを SXF 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL **IKSxfP21FileSave**(LPCTSTR FileName, HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
BOOL **IKSxfSfcFileSave**(LPCTSTR FileName, HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function **IKSxfP21FileSave**(FileName: PChar; Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;
function **IKSxfSfcFileSave**(FileName: PChar; Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ベクトルイメージのメモリハンドル
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
p21 形式のファイルに保存する場合は **IKSxfP21FileSave** 関数を、sfc 形式のファイルに保存する場合は **IKSxfSfcFileSave** 関数を使用します。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合
IKSxfP21FileSave("c:¥¥abc.p21",・・・・・);
(2)FTP サーバーにファイルを保存する場合
IKSxfP21FileSave("ftp://www.newtone.co.jp/image/abc.p21;;;true;user;password",・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無("true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。
(3)HTTP(S)サーバーにファイルを保存する場合
IKSxfP21FileSave("http://www.newtone.co.jp/image/abc.p21;;;;user;password",・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略すると HTTP サーバからの処理となります。

【ImageKit7 との違い】
FTP サーバーや HTTP(S)サーバーに直接ファイルを保存することができるようになりました。

IKSxfP21FileSaveMem,IKSxfSfcFileSaveMem

【機能】
ベクトルイメージを SXF 形式で Raw データに保存します。

【関数書式】
(1)C++Builder
BOOL **IKSxfP21FileSaveMem**(HANDLE InHandle, HANDLE *OutHandle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
BOOL **IKSxfSfcFileSaveMem**(HANDLE InHandle, HANDLE *OutHandle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function **IKSxfP21FileSaveMem**(InHandle: THandle; var OutHandle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;
function **IKSxfSfcFileSaveMem**(InHandle: THandle; var OutHandle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	ベクトルイメージのメモリハンドル
OutHandle	保存する Raw データ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
p21 形式の Raw データに保存する場合は **IKSxfP21FileSaveMem** 関数を、sfc 形式の Raw データに保存する場合は **IKSxfSfcFileSaveMem** 関数を使用します。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKSxfP21FileSave**,**IKSxfSfcFileSave** 関数と同じです。当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

(注意)
内部で一時的にテンポラリファイルを作成します。

IKTiffFileLoad

【機能】
TIFF 形式のファイルからラスタイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKTiffFileLoad(LPCTSTR FileName, int Page, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKTiffFileLoad(FileName: PChar; Page: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
Page	読み込むファイルのページ番号(0～)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
複数のイメージが存在する場合は、Page で指定されたイメージを読み込みます。1 イメージしかないファイルは Page に指定した値は無効となります。対応イメージは 1,4,8,16,24,32 ビットカラーです(圧縮モードにより異なります)。イメージファイルのページ数は **IKFileType** 関数で取得できます。
TIFF の JPEG 形式については読み込みに失敗するファイルも存在しますのでご了承ください。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

IKTiffFileLoadMem

【機能】

TIFF 形式の Raw データからラスタイメージをメモリ上に読み込みます。

【関数書式】

(1)C++Builder

HANDLE IKTiffFileLoadMem(HANDLE Handle, int Page, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);

(2)Delphi

function IKTiffFileLoadMem(Handle: THandle; Page: Integer; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	TIFF 形式の Raw データ
Page	読み込むファイルのページ番号(0～)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】

ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】

複数のイメージが存在する場合は、Page で指定されたイメージを読み込みます。1 イメージしかない Raw データは Page に指定した値は無効となります。対応イメージは 1,4,8,16,24,32 ビットカラーです(圧縮モードにより異なります)。イメージファイルのページ数は **IKFileType** 関数で取得できます。

TIFF の JPEG 形式については読み込みに失敗する場合がありますのでご了承ください。

ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKTiffFileLoad** 関数と同じです。

IKTiffFileSave

【機能】
ラスタイメージを TIFF 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKTiffFileSave(LPCTSTR FileName, HANDLE Handle, int Type, int Color, BOOL Append, LPCSTR Comment, BOOL OneStrip, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKTiffFileSave(FileName: PChar; Handle: THandle; Type_, Color: Integer; Append: LongBool; Comment: PAnsiChar; OneStrip: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ラスタイメージのメモリハンドル
Type	保存するファイルタイプ 0:非圧縮　1:CCITTRLE　2:GROUP3 1D　3:GROUP3 2D 4:GROUP4　5:PACKBITS　6:LZW　7:JPEG
Color	Type が 0,5,6 で保存対象イメージが 24,32 ビットカラーの場合　0:RGB, 1:CMYK Type が 7 の場合　JPEG 圧縮品質係数(0～100/推奨 75) Type が 1,2,3,4 の場合は無効
Append	追加保存の設定(False(0):設定なし、True(0 以外):設定あり) True(0 以外)を設定するとファイルの最後のページに該当するイメージを付加します。
Comment	保存するコメント文字列(MAX:1023 バイト) 次のように設定してください。「Document + 0x0d + Description + ・・・　+ 0x0d」 Document:　文書名 Description:　画像説明 Page:　ページ名 Software:　ソフトウエア DateTime:　日時 Artist:　作者 ※16 進表記のため、Delphi は 0x を$に置き換えてください。
OneStrip	ストリップの設定(False(0):複数のストリップ、True(0 以外):1 つのストリップ)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対応イメージは 1,4,8,16(グレーは除く),24,32 ビットカラーです。(圧縮モードにより異なります。)
Type が 0,5,6 のいずれかで、保存対象のイメージが 24,32 ビットカラーで Color が 1(CMYK)の場合は、出力されるファイルのビット数は 32 となります。
OneStrip は GROUP4 以外の形式で保存する場合に有効です。(GROUP4 は 1 ストリップ固定です。)
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
TIFF の GROUP3-1D は MH、GROUP3-2D は MR、GROUP4 は MMR と同じ形式です。

TIFF の非圧縮、PACKBITS、LZW 形式で 16 ビットイメージを保存すると 24 ビットイメージに変換されます。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合
IKTiffFileSave("c:¥¥abc.tif",・・・・・);
(2)FTP サーバーにファイルを保存する場合
IKTiffFileSave("ftp://www.newtone.co.jp/image/abc.tif;;;true;user;password",・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無("true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。
(3)HTTP(S)サーバーにファイルを保存する場合
IKTiffFileSave("http://www.newtone.co.jp/image/abc.tif;;;;user;password",・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略すると HTTP サーバからの処理となります。

【ImageKit5 との違い】

関数名	**引数の並び**
IK5TiffFileSave:	FileName, Handle, Type, Color, Append, Comment, FileUserProc, Caption, Message, Button
IKTiffFileSave:	FileName, Handle, Type, Color, Append, Comment, OneStrip, UserProc, Caption, Message, Button

OneStrip が引数に追加されました。FileUserProc と UserProc は同じユーザ関数を表します。OneStrip を False(0)にすると IK5 と同様に動作します。

【ImageKit7 との違い】
FTP サーバーや HTTP(S)サーバーに直接ファイルを保存することができるようになりました。

IKTiffFileSaveMem

【機能】
ラスタイメージを TIFF 形式で Raw データに保存します。

【関数書式】
(1)C++Builder
BOOL IKTiffFileSaveMem(HANDLE InHandle, HANDLE *OutHandle, int Type, int Color, BOOL Append, LPCSTR Comment, BOOL OneStrip, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKTiffFileSaveMem(InHandle: THandle; var OutHandle: THandle; Type_, Color: Integer; Append: LongBool; Comment: PAnsiChar; OneStrip: LongBool; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	ラスタイメージのメモリハンドル
OutHandle	保存する Raw データ
Type	保存するファイルタイプ 0:非圧縮　1:CCITTRLE　2:GROUP3 1D　3:GROUP3 2D 4:GROUP4　5:PACKBITS　6:LZW　7:JPEG
Color	Type が 0,5,6 で保存対象イメージが 24,32 ビットカラーの場合　0:RGB, 1:CMYK Type が 7 の場合　JPEG 圧縮品質係数(0～100/推奨 75) Type が 1,2,3,4 の場合は無効
Append	追加保存の設定(False(0):設定なし、True(0 以外):設定あり) True(0 以外)を設定するとファイルの最後のページに該当するイメージを付加します。
Comment	保存するコメント文字列(MAX:1023 バイト) 次のように設定してください。「Document + 0x0d + Description + ・・・　+ 0x0d」 Document:　文書名 Description:　画像説明 Page:　ページ名 Software:　ソフトウエア DateTime:　日時 Artist:　作者 ※16 進表記のため、Delphi は 0x を$に置き換えてください。
OneStrip	ストリップの設定(False(0):複数のストリップ、True(0 以外):1 つのストリップ)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
対応イメージは 1,4,8,16(グレーは除く),24,32 ビットカラーです。(圧縮モードにより異なります。)
Type が 0,5,6 のいずれかで、保存対象のイメージが 24,32 ビットカラーで Color が 1(CMYK)の場合は、出力されるファイルのビット数は 32 となります。
OneStrip は GROUP4 以外の形式で保存する場合に有効です。(GROUP4 は 1 ストリップ固定です。)
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については、**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。

TIFF の GROUP3-1D は MH、GROUP3-2D は MR、GROUP4 は MMR と同じ形式です。
TIFF の非圧縮、PACKBITS、LZW 形式で 16 ビットイメージを保存すると 24 ビットイメージに変換されます。
ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKTiffFileSave** 関数と同じです。

IKWmfFileLoad

【機能】
WMF 形式のファイルからベクトルイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKWmfFileLoad(LPCTSTR FileName, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKWmfFileLoad(FileName: PChar; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
FileName	読み込むファイル名
Width	読み込むイメージの幅(ピクセル)
Height	読み込むイメージの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Width もしくは Height が 0 以下の場合は WMF ファイルに格納されているサイズで読み込みます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

(注意)
読み込まれるイメージの縦横のサイズは、設定した矩形内(Width * Height)で縦横比を考慮した最も大きなサイズとなりますので、設定したサイズ通りにならない場合があります。

IKWmfFileLoadMem

【機能】
WMF 形式の Raw データからベクトルイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKWmfFileLoadMem(HANDLE Handle, long Width, long Height, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKWmfFileLoadMem(Handle: THandle; Width, Height: Longint; UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	WMF 形式の Raw データ
Width	読み込むイメージの幅(ピクセル)
Height	読み込むイメージの高さ(ピクセル)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
Width もしくは Height が 0 以下の場合は WMF ファイルに格納されているサイズで読み込みます。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データから読み込む違いはありますが、動作としては **IKWmfFileLoad** 関数と同じです。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

(注意)
読み込まれるイメージの縦横のサイズは、設定した矩形内(Width * Height)で縦横比を考慮した最も大きなサイズとなりますので、設定したサイズ通りにならない場合があります。

IKWmfFileSave

【機能】
ベクトルイメージを WMF 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKWmfFileSave(LPCTSTR FileName, HANDLE Handle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKWmfFileSave(FileName: PChar; Handle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
FileName	保存するファイル名
Handle	ベクトルイメージのメモリハンドル
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

FileName に FTP サーバーの名称や HTTP(S)サーバーの名称などを設定すると FTP サーバーや HTTP(S)サーバーに直接ファイルを保存できます。

(1)ローカルドライブやネットワークドライブにファイルを保存する場合
IKWmfFileSave("c:¥¥abc.wmf",・・・・・);
(2)FTP サーバーにファイルを保存する場合
IKWmfFileSave("ftp://www.newtone.co.jp/image/abc.wmf;;;true;user;password",・・・・・);

(*)FileName を設定する際の順番
FTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;パッシブモード;ユーザ名;パスワード

プロキシサーバを設定したり、ポート番号やパッシブ(PASV)モード接続を変更する場合はプロパティに「FTP サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;パッシブモード接続の有無」を渡してください。(文字列をセミコロンで区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、パッシブモード接続の有無("true"または"false" - 大小文字関係なし)を省略するとパッシブモード接続となります。
(3)HTTP(S)サーバーにファイルを保存する場合
IKWmfFileSave("http://www.newtone.co.jp/image/abc.wmf;;;;user;password",・・・・・);

(**)FileName を設定する際の順番
HTTP://サーバ名/フォルダ名/ファイル名;プロキシ名;ポート番号;HTTPS;ユーザ名;パスワード

プロキシサーバを設定したりポート番号を変更、または HTTPS を利用する場合はプロパティに「HTTP(S)サーバの名称もしくは IP アドレス;プロキシサーバの名称または IP アドレス;ポート番号;HTTPS」を渡してください。(文字列をセミコロンで

区切る)
なお、ポート番号を省略するとデフォルトポートが割り当てられ、"HTTPS"を省略するとHTTPサーバからの処理となります。

【ImageKit7との違い】
FTPサーバーやHTTP(S)サーバーに直接ファイルを保存することができるようになりました。

IKWmfFileSaveMem

【機能】
ベクトルイメージを WMF 形式で Raw データに保存します。

【関数書式】
(1)C++Builder
BOOL IKWmfFileSaveMem(HANDLE InHandle, HANDLE *OutHandle, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKWmfFileSaveMem(InHandle: THandle; var OutHandle: THandle; UserProc: LONG_PTR; Caption, Message, Button: PChar): LongBool;

【引数】

名称	内容
InHandle	ベクトルイメージのメモリハンドル
OutHandle	保存する Raw データ
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については**「Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll」のユーザ関数の定義**をご覧ください。
ファイルあるいは Raw データに保存する違いはありますが、動作としては **IKWmfFileSave** 関数と同じです。
当関数を実行する場合は、事前に **Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll** の **IKVectorGdipStart** 関数を実行する必要があります。

IKYCCBmpPlaneFileLoad

【機能】
YCrCb プレーン毎に保存してある BMP ファイルからラスタイメージをメモリ上に読み込みます。

【関数書式】
(1)C++Builder
HANDLE IKYCCBmpPlaneFileLoad(LPCTSTR YFileName, LPCTSTR CrFileName, LPCTSTR CbFileName);
(2)Delphi
function IKYCCBmpPlaneFileLoad(YFileName, CrFileName, CbFileName: PChar): THandle;

【引数】

名称	内容
YFileName	Y プレーンとして読み込むファイル名
CrFileName	Cr プレーンとして読み込むファイル名
CbFileName	Cb プレーンとして読み込むファイル名

【戻り値】
ラスタイメージのメモリハンドル（実行に失敗した場合は 0 が返されます）。

【解説】
YCrCb それぞれのプレーンを表すイメージは 8 ビットグレーで、出力イメージは 24 ビットカラーです。

IKYCCBmpPlaneFileSave

【機能】
ラスタイメージを YCrCb プレーン毎に BMP 形式でファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKYCCBmpPlaneFileSave(LPCTSTR YFileName, LPCTSTR CrFileName, LPCTSTR CbFileName, HANDLE Handle);
(2)Delphi
function IKYCCBmpPlaneFileSave(YFileName, CrFileName, CbFileName: PChar; Handle: THandle): LongBool;

【引数】

名称	内容
YFileName	Y プレーンとして保存するファイル名
CrFileName	Cr プレーンとして保存するファイル名
CbFileName	Cb プレーンとして保存するファイル名
Handle	ラスタイメージのメモリハンドル

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
保存対象イメージは 24 ビットカラーです。Handle を YCrCb のそれぞれのプレーン毎にファイルに保存します。保存されるイメージは 8 ビットグレーとなります。

1－4. Ik10Print.dll/Ik10PrintA.dll/Ik10Print64.dll/Ik10Print64A.dll

描画先オブジェクト(スクリーン、プリンタ、メモリハンドル)に対して図形や文字などを描画する機能やプリンタを制御する機能を提供します。

描画先オブジェクト:S=スクリーン、P=プリンタ、M=メモリハンドル

●DLL 関数コマンド一覧(アルファベット順)

関数名	内容
IKArc	弧の描画(S,P,M)
IKChord	弓形の描画(S,P,M)
IKDrawFocusRect	フォーカス付き矩形枠の描画(S)
IKDrawString	GDI+を使用したテキストの描画(S,P,M)
IKDrawText	矩形内にテキストを描画(S,P,M)
IKEllipse	矩形に内接する円の描画(S,P,M)
IKEnumPaperBins	プリンタの用紙トレイ名リストを取得
IKEnumPaperSizes	プリンタの用紙サイズ名リストを取得
IKEnumPorts	ポートの列挙
IKEnumPrinters	インストールされているプリンタの列挙
IKEnumResolutions	プリンタの解像度リストを取得
IKFillRect	矩形の塗りつぶし(S,P,M)
IKFrameRect	ブラシによる矩形の描画(S,P,M)
IKGetDefaultPrinter	通常使うプリンタの取得
IKGetDevModeHandle	指定したプリンタの DEVMODE 構造体のハンドルを取得
IKGetDevModeHandleEx	指定したプリンタの DEVMODE 構造体と DEVNAMES 構造体のハンドルを取得
IKGetDevModeInfo	DEVMODE 構造体のハンドルから印刷情報の取得
IKGetDevNamesInfo	DEVNAMES 構造体のハンドルから文字列情報の取得
IKGetImageFromHdc	デバイスコンテキスト(hDC)からイメージを取得
IKGetPaperSize	指定したプリンタの用紙の有効印字領域、高さ、幅を取得
IKGetPixel	指定したピクセルから RGB 値を取得(S,M)
IKGetPrinterPort	プリンタのポートの取得
IKGetTextExtent	指定した文字情報から描画する文字列の高さと幅を取得(S,P)
IKImageOut	デバイスコンテキスト(hDC)に対するイメージの描画(S,P)
IKImageOutToHwnd	ウィンドウハンドル(hWnd)に対するイメージの描画(S)
IKLine	直線の描画(S,P,M)
IKMeasureString	指定した文字情報から描画する文字列の高さと幅を取得(GDI+) (S,P)
IKPaint	指定した色を別の色に設定(S,P,M)
IKPie	扇形の描画(S,P,M)
IKPolyBezier	ベジェ曲線の描画(S,P,M)
IKPolygon	多角形の描画(S,P,M)
IKPolyline	連続線の描画(S,P,M)
IKPreviewInit	印刷プレビュー時の DPI を設定
IKPrintAbortDoc	印刷ジョブの終了
IKPrintCreateDC	デバイスコンテキストの作成
IKPrintCreateDCEx	デバイスコンテキストの作成
IKPrintDeleteDC	デバイスコンテキストの削除
IKPrintDialog	デバイスコンテキストの作成(印刷ダイアログ表示)
IKPrintDlg	デバイスコンテキストの作成(印刷ダイアログ表示)
IKPrintEndDoc	プリントの終了
IKPrintEndPage	ページの終了
IKPrintGdipEnd	GDI+の終了処理
IKPrintGdipStart	GDI+の開始処理
IKPrintGetArrayNum	列挙する情報に応じて必要な配列の要素数を取得
IKPrintStartDoc	プリントの開始
IKPrintStartPage	ページの開始
IKRectangle	矩形の描画(S,P,M)

IKReleaseDevModeHandle	DEVMODE 構造体や DEVNAMES 構造体のハンドルを解放
IKReleaseDevModeHandleEx	DEVMODE 構造体と DEVNAMES 構造体のハンドルを解放
IKRoundRect	角が丸い矩形の描画(S,P,M)
IKSaveDevModeHandle	DEVMODE 構造体のハンドルをファイルに保存
IKSaveDevModeHandleEx	DEVMODE 構造体と DEVNAMES 構造体のハンドルをファイルに保存
IKSetDefaultPrinter	通常使うプリンタを設定
IKSetDevModeInfo	DEVMODE 構造体のハンドルへ印刷情報を設定
IKSetPixel	指定したピクセルに RGB 値を設定(S,M)
IKSetPrint	プリンタの情報を設定ファイルに保存
IKTextOut	始点を与えてテキストを描画(S,P,M)

●構造体(ユーザ定義型)の定義

IKPRINT_DEVMODEINFO: 印刷情報を取得したり、設定する時に使用します。

(1)C++Builder

```
typedef struct {
        short           Orientation;
        short           PaperSize;
        short           Zoom;
        WORD            Copies;
        short           PaperBin;
        short           XResolution;
        short           YResolution;
        short           ColorMode;
        short           Duplex;
        BYTE            Collate;
        short           CustomPaperWidth;
        short           CustomPaperHeight;
} IKPRINT_DEVMODEINFO;
```

(2)Delphi

```
type
  IKPRINT_DEVMODEINFO = Record
    Orientation:        Smallint;
    PaperSize:          Smallint;
    Zoom:               Smallint;
    Copies:             Word;
    PaperBin:           Smallint;
    XResolution:        Smallint;
    YResolution:        Smallint;
    ColorMode:          Smallint;
    Duplex:             Smallint;
    Collate:            Byte;
    CustomPaperWidth:   Smallint;
    CustomPaperHeight:  Smallint;
  end;
```

Orientation: 印刷の向き 1: 縦(DMORIENT_PORTRAIT), 2: 横(DMORIENT_LANDSCAPE)

PaperSize: 用紙サイズ

1: レター、8.5 x 11 インチ (DMPAPER_LETTER)
2: レター スモール、8.5 x 11 インチ (DMPAPER_LETTERSMALL)
3: タブロイド、11 x 17 インチ (DMPAPER_TABLOID)
4: レジャー、17 x 11 インチ (DMPAPER_LEDGER)
5: リーガル、8.5 x 14 インチ (DMPAPER_LEGAL)
6: ステートメント、5.5 x 8.5 インチ (DMPAPER_STATEMENT)
7: エグゼクティブ、7.25 x 10.5 インチ (DMPAPER_EXECUTIVE)
8: A3、297 x 420 mm (DMPAPER_A3)
9: A4、210 x 297 mm (DMPAPER_A4)
10: A4 Small、210 x 297 mm (DMPAPER_A4SMALL)

11: A5、148 x 210 mm (DMPAPER_A5)
12: B4、250 x 354 mm (DMPAPER_B4)
13: B5、182 x 257 mm (DMPAPER_B5)
256: ユーザ定義サイズ(DMPAPER_USER)

Zoom: スケール時の百分率(0～100)
Copies: 印刷部数(1～)
PaperBin: 用紙トレイ
1: 上段用紙トレイ(DMBIN_UPPER)
2: 下段用紙トレイ(DMBIN_LOWER)
3: 中段用紙トレイ(DMBIN_MIDDLE)
4: 手差し用紙フィーダ(DMBIN_MANUAL)
5: 封筒フィーダ(DMBIN_ENVELOPE)
6: 手差し封筒フィーダ(DMBIN_ENVMANUAL)
7: 自動用紙トレイ選択(DMBIN_AUTO)
8: トラクタフィーダ(DMBIN_TRACTOR)
9: 小判用紙ソース(DMBIN_SMALLFMT)
10: 大判用紙ソース(DMBIN_LARGEFMT)
11: 大容量の用紙トレイ(DMBIN_LARGECAPACITY)
14: 用紙カセット(DMBIN_CASSETTE)
XResolution: プリンタの X 方向の解像度 *1
正の値: 1 インチあたりのドット数(DPI)
-1: ドラフトの印刷解像度(DMRES_DRAFT)
-2: 低解像度(DMRES_LOW)
-3: 中解像度(DMRES_MEDIUM)
-4: 高解像度(DMRES_HIGH)
YResolution: プリンタの Y 方向の解像度(0～) *2
ColorMode: カラープリンタでカラーを使用するかモノクロを使用するかを設定します。
1: モノクロ(DMCOLOR_MONOCHROME), 2: カラー(DMCOLOR_COLOR)
Duplex: 両面印刷をサポートしている場合、両面印刷を行うかどうかを設定します。
1: 両面印刷なし(DMDUP_SIMPLEX)
2: 垂直(短い軸)方向の両面印刷(DMDUP_VERTICAL)
3: 水平(長い軸)方向の両面印刷(DMDUP_HORIZONTAL)
Collate: 0 以外:部単位で印刷
CustomPaperWidth: **PaperSize** = 256 の時の用紙の横幅(0.1mm 単位) *2
CustomPaperHeight: **PaperSize** = 256 の時の用紙の縦幅(0.1mm 単位) *2

※()内の説明は WindowsAPI で使用する定数と同じ意味です。

PaperSize については 14 以降の値も取得および設定可能です。詳しくはご利用のコンテナの WindowsAPI に関連する部分を参考してください。
各項目に値を設定する場合、プリンタドライバに依存します。そのため、設定しても効果がまったくなかったり、異なる設定を行っても印刷結果が同じになることがあります。詳しくはプリンタドライバのマニュアルなどを参照してください。

*1 名称が変更されました(PrintQuality → XResolution)。
*2 ImageKit7 で追加されたメンバー変数です
ImageKit6 から移行する場合は注意してください。

IKPRINT_DIALOG: 印刷ダイアログ(**IKPrintDialog, IKPrintDlg**)で使用します。

(1)C++Builder

```
typedef struct {
    BYTE        Collate;
    WORD        Copies;
    WORD        FromPage;
    WORD        ToPage;
    WORD        MaxPage;
    WORD        MinPage;
    DWORD       Options;
    BYTE        PrintRange;
```

```
    BYTE            PrintToFile;
} IKPRINT_DIALOG;
typedef IKPRINT_DIALOG * PTR_IKPRINT_DIALOG;
```

(2)Delphi

```
type
  IKPRINT_DIALOG = Record
    Collate:        Byte;
    Copies:         Word;
    FromPage:       Word;
    ToPage:         Word;
    MaxPage:        Word;
    MinPage:        Word;
    Options:        DWORD;
    PrintRange:     Byte;
    PrintToFile:    Byte;
  end;
```

Collate: [印刷]ダイアログの[部単位で印刷](0 以外:部単位で印刷にチェック)
Copies : 印刷部数(1～)
FromPage: 印刷開始ページ(1～)
ToPage: 印刷終了ページ(1～)
MaxPage: 印刷最大ページ(1～)
MinPage: 印刷最小ページ(1～)
Options: [印刷]ダイアログの初期設定
0x4 [選択された部分]ラジオボタンを使用禁止(PD_NOSELECTION)
0x8 [ページ指定]ラジオボタンを使用禁止(PD_NOPAGENUMS)
0x80 デフォルトプリンタがない場合に警告メッセージを非表示(PD_NOWARNING)
0x800 ヘルプボタンを表示(PD_SHOWHELP)
0x80000 [ファイルへ出力]チェックボックスを使用禁止(PD_DISABLEPRINTTOFILE)
0x100000 [ファイルへ出力]チェックボックスを非表示(PD_HIDEPRINTTOFILE)
・16 進表記のため、Delphi は 0x を$に変換してください。
・()内の説明は WindowsAPI で使用する定数と同じ意味です。
・複数の項目を指定する場合は｜,or,Or で組み合わせてください。
PrintRange: 印刷範囲の種類(0:すべて,1:選択した部分,2:ページ指定)
PrintToFile: [印刷]ダイアログの[ファイルへ出力](0 以外:ファイルへ出力にチェック)

IKPRINT_DRAWINFO: 図形を描画する際に使用します。

(1)C++Builder

```
typedef struct {
    long            PenWidth;
    BYTE            PenStyle;
    BYTE            PenMode;
    COLORREF        PenColor;
    BYTE            BrushStyle;
    COLORREF        BrushColor;
    BYTE            Transparent;
    COLORREF        BackColor;
} IKPRINT_DRAWINFO;
typedef IKPRINT_DRAWINFO * PTR_IKPRINT_DRAWINFO;
```

(2)Delphi

```
type
  IKPRINT_DRAWINFO = Record
    PenWidth:       Longint;
    PenStyle:       Byte;
    PenMode:        Byte;
    PenColor:       COLORREF;
```

```
    BrushStyle:         Byte;
    BrushColor:         COLORREF;
    Transparent:        Byte;
    BackColor:          COLORREF;
  end;
```

PenWidth は、ペンの幅を指定します。
描画対象がスクリーンとメモリハンドルの場合はピクセル単位で、プリンタの場合は 0.1mm 単位で設定してください。

PenStyle は、描画するペンのスタイル指定します。
値の説明は以下に示します。()内の説明は WindowsAPI で使用する定数と同じ意味です。
0:線は描画されない(PS_NULL)
1:実線(PS_SOLID)
2:破線(PS_DASH)
3:点線(PS_DOT)
4:一点鎖線(PS_DASHDOT)
5:二点鎖線(PS_DASHDOTDOT)
6:実線(PS_INSIDEFRAME)

注意:
IKLine、IKRectangle 以外のコマンドで図形を描画する場合に、**PenWidth** に 1 以上(ピクセルで与えた場合、0.1mm の場合もピクセルに換算して判定)、**PenStyle** で 2～5 のいずれかを設定した場合には **PenWidth** を強制的に 1 とします。

PenMode は、描かれるペンの色をキャンバス上の色とどのように対応させるかを指定します。
値の説明は以下に示します。()内の説明は WindowsAPI で使用する定数と同じ意味です。
1:黒(R2_BLACK)
2:ペン色と画面色の組み合わせの反転(R2_NOTMERGEPEN)
3:画面色とペン反転色のどちらにも共通な色の組み合わせ(R2_MASKNOTPEN)
4:ペンの反転色(R2_NOTCOPYPEN)
5:ペン色と画面反転色のどちらにも共通な色の組み合わせ(R2_MASKPENNOT)
6:画面色の反転色(R2_NOT)
7:ペン色または画面色の(両方ではなく)どちらかの色の組み合わせ(R2_XORPEN)
8:ペン色と画面色のどちらにも共通な色の組み合わせの反転(R2_NOTMASKPEN)
9:ペン色と画面色のどちらにも共通な色の組み合わせ(R2_MASKPEN)
10:ペン色または画面色の(両方ではなく)どちらかの色の組み合わせの反転(R2_NOTXORPEN)
11:変化なし(R2_NOP)
12:画面色とペン反転色の組み合わせ(R2_MERGENOTPEN)
13:**PenColor** で指定したペン色(R2_COPYPEN)、デフォルト値
14:ペン色と画面反転色の組み合わせ(R2_MERGEPENNOT)
15:ペン色と画面色の組み合わせ(R2_MERGEPEN)
16:白(R2_WHITE)

PenColor は、描画するペンの色を指定します。値を設定する場合は、**RGB**(Red,Green,Blue)の戻り値などを与えます。
(デバイスが設定した色を正確に表現できない場合は、近似の色が用いられます。)

BrushStyle は、塗りつぶすブラシの模様を指定します。
値の説明は以下に示します。()内の説明は WindowsAPI で使用する定数と同じ意味です。
0:塗りつぶしは行わない(BS_NULL)
1:塗りつぶし(BS_SOLID)
2:左下から右上へ 45 度の角度で向かう斜線のハッチパターン(BS_HATCHED、HS_BDIAGONAL)
3:縦横の格子線のハッチパターン(BS_HATCHED、HS_CROSS)
4:3 のハッチパターンを 45 度回転したもの(BS_HATCHED、HS_DIAGCROSS)
5:左上から右下へ 45 度の角度で向かう斜線のハッチパターン(BS_HATCHED、HS_FDIAGONAL)
6:横線のハッチパターン(BS_HATCHED、HS_HORIZONTAL)
7:縦線のハッチパターン(BS_HATCHED、HS_VERTICAL)

BrushColor は、塗りつぶすブラシの色を指定します。値を設定する場合は、**RGB**(Red,Green,Blue)の戻り値などを与えます。
(デバイスが設定した色を正確に表現できない場合は、近似の色が用いられます。)

Transparent は、背景や線と線の間隔を描画する時の透過を指定します。(透過にする場合は 0 以外を設定します。)

BackColor: *1
ペンが実線以外、あるいはブラシがハッチパターンの場合に線と線の間の色および背景色を指定します。値を設定する場合は、**RGB**(Red,Green,Blue)の戻り値などを与えます。(デバイスが設定した色を正確に表現できない場合は、近似の色が用いられます。)
Transparent = 0 の場合に有効です。

*1 ImageKit7 で追加されたメンバー変数です。ImageKit5/6 から移行する場合は注意してください。

IKPRINT_TEXTINFO: 文字を描画する際に使用します。

(1)C++Builder

```
typedef struct {
        short           CharSet;
        TCHAR           FontName[LF_FACESIZE];
        WORD            FontSize;
        BYTE            FontBold;
        BYTE            FontItalic;
        BYTE            FontUnderline;
        BYTE            FontStrikeOut;
        BYTE            Transparent;
        BYTE            Direction;
        short           CharAngle;
        long            CharExtra;
        BYTE            HCentering;
        BYTE            VCentering;
        BYTE            RotateString;
        COLORREF        TextColor1;
        COLORREF        TextColor2;
        BYTE            Alpha1;
        BYTE            Alpha2;
        WORD            HotkeyPrefix;
} IKPRINT_TEXTINFO;
typedef IKPRINT_TEXTINFO * PTR_IKPRINT_TEXTINFO;
```

(2)Delphi

```
type
  IKPRINT_TEXTINFO = Record
    CharSet:        Smallint;
    FontName:       array [0..LF_FACESIZE - 1] of Char;
    FontSize:       Word;
    FontBold:       Byte;
    FontItalic:     Byte;
    FontUnderline:  Byte;
    FontStrikeOut:  Byte;
    Transparent:    Byte;
    Direction:      Byte;
    CharAngle:      Smallint;
    CharExtra:      Longint;
    HCentering:     Byte;
    VCentering:     Byte;
    RotateString:   Byte;
    TextColor1:     COLORREF;
    TextColor2:     COLORREF;
    Alpha1:         Byte;
    Alpha2:         Byte;
    HotkeyPrefix:   Word;
  end;
```

CharSet は、フォントで使用する文字セットを指定します。0～255 の値を設定した場合は、Windows で定義されている値と同じになります。
　-1:ANSI 文字 or 日本語シフト JIS 文字［ImageKit5 互換］
　 0:ANSI 文字(ANSI_CHARSET)
　 1:FontName と FontSize により選択される(DEFAULT_CHARSET)
　 2:シンボル文字(SYMBOL_CHARSET)
　128:日本語シフト JIS 文字(SHIFTJIS_CHARSET)
　255:オペレーティングシステムに依存(OEM_CHARSET)
※()内の説明は WindowsAPI で使用する定数と同じ意味です。

FontName は、テキストを描画する際のフォントの名称を指定します。(例えば、"MSゴシック"など)
FontSize は、テキストを描画する際のフォントのサイズをポイント数で指定します。
FontBold は、フォントを太字にするかしないかを指定します。(太字にする場合は 0 以外を設定します。)
FontItalic は、フォントを斜体にするかしないかを指定します。(斜体にする場合は 0 以外を設定します。)
FontUnderline は、テキストに下線を付加するかしないかを指定します。(下線を付加する場合は 0 以外を設定します。ただし、0 以外でも **Direction** や **CharAngle** の値により無効となる場合があります。)
FontStrikeOut は、テキストに打ち消し線を付加するかしないかを指定します。(打ち消し線を付加する場合は 0 以外を設定します。ただし、0 以外でも **Direction** や **CharAngle** の値により無効となる場合があります。)
Transparent は、テキストの背景を描画する時の透過を指定します。(透過にする場合は 0 以外を設定します。その場合は、**TextColor2** に値を設定しても無視されます。)

Direction は、文字列の描画方向を指定します。(**RotateString** が False の場合に有効)
　0:左から右(横書き)、1:右から左(横書き)、2:上から下(縦書き)、3:下から上(縦書き)
CharAngle は、文字の回転角度を 1 度単位で指定します。(0～360)
　IKDrawText,IKTextOut は反時計回りに、IKDrawString は時計回りに回転
CharExtra は、描画する文字の間隔を指定します。(0～)
　描画対象がスクリーンとメモリハンドルの場合はピクセル単位で、プリンタの場合は 0.1mm 単位で設定してください。
HCentering は、文字列の水平センタリングを指定します。(IKDrawString,IKDrawText,IKMeasureString で使用します。)
　0:左、1:中央、2:右
VCentering は、文字列の垂直センタリングを指定します。(IKDrawString,IKDrawText,IKMeasureString で使用します。)
　0:上、1:中央、2:下
RotateString は、文字単位で回転するか文字列単位で回転するかを指定します。(IKGetTextExtent,IKTextOut で使用します。)
　0:文字単位(**CharAngle** は 0,90,180,270 のみ有効)、0 以外:文字列単位

TextColor1: *1
　IKDrawText,IKTextOut ではテキストの描画色を指定します。
　IKDrawString ではブラシの種類により意味合いが異なります。
　　ソリッドブラシ:ブラシの色
　　ハッチブラシ:描画される線の色
　　グラデーションブラシ:線形グラデーションの開始色
　　※テクスチャブラシでは無効
TextColor2: *1
　IKDrawText,IKTextOut ではテキストの背景色を指定します。
　IKDrawString ではブラシの種類により意味合いが異なります。
　　ハッチブラシ:線間の領域の色
　　グラデーションブラシ:線形グラデーションの終了色
　　※ソリッドブラシとテクスチャブラシでは無効
値を設定する場合は、**RGB**(Red,Green,Blue)などの戻り値を与えます。
(デバイスが設定した色を正確に表現できない場合は、近似の色が用いられます。)

Alpha1 は **TextColor1** のアルファ値です。255 の場合、完全不透明です。(IKDrawString で使用します。) *2
Alpha2 は **TextColor2** のアルファ値です。255 の場合、完全不透明です。(IKDrawString で使用します。) *2

HotkeyPrefix はプリフィックス文字をどう扱うかを指定します。 *2
　IKDrawText の場合 0:プリフィックスなし,1:プリフィックス表示
　IKDrawString の場合 0:プリフィックスなし, 1:プリフィックス表示, 2:プリフィックス非表示

*1 名称が変更されました。
TextForeColor → TextColor1
TextBackColor → TextColor2
*2 ImageKit7 で追加されたメンバー変数です。
ImageKit5/6 から移行する場合は注意してください。

IKArc

【機能】
描画先オブジェクトに弧を描画します。

【関数書式】
(1)C++Builder
BOOL IKArc(LPVOID DeviceValue, LPRECT PrintRect, int x1, int y1, int x2, int y2, PTR_IKPRINT_DRAWINFO DrawInfo, BYTE DeviceMode);
(2)Delphi
function IKArc(DeviceValue: THandle; var PrintRect: TRect; x1, y1, x2, y2: Integer; var DrawInfo: IKPRINT_DRAWINFO; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
PrintRect	境界矩形を指定する構造体変数
x1,y1	弧の始点の x,y 座標
x2,y2	弧の終点の x,y 座標
DrawInfo	描画する情報を指定する構造体変数
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PrintRect で指定した矩形に含まれる楕円弧を 2 点間で描画します。始点と終点が弧の上にある必要はなく、指定の点から境界矩形までの中心までの直線が計算され、弧とその直線との交点が用いられます。
(描画対象はスクリーン、プリンタ、メモリハンドル)
描画するには DrawInfo の **PenWidth,PenStyle,PenMode,PenColor,Transparent,BackColor** に値を設定する必要があります。
BackColor は **Transparent** = 0 でペンが実線以外の場合に有効です。

PrintRect のメンバー変数と x1,y1,x2,y2 の値をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
PrintRect のメンバー変数と x1,y1,x2,y2 の値を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5Arc:	DeviceValue, PrintRect, x1, y1, x2, y2, PrintInfo
IKArc:	DeviceValue, PrintRect, x1, y1, x2, y2, DrawInfo, DeviceMode

PrintInfo と DrawInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。Transparent を 0 以外に設定し、IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

【ImageKit6 との違い】
IKPRINT_DRAWINFO 構造体に BackColor が追加されました。Transparent が 0 以外であれば IK6 と同じ動作です。

IKChord

【機能】
描画先オブジェクトに弓形を描画します。

【関数書式】
(1)C++Builder
BOOL IKChord(LPVOID DeviceValue, LPRECT PrintRect, int x1, int y1, int x2, int y2, PTR_IKPRINT_DRAWINFO DrawInfo, BYTE DeviceMode);
(2)Delphi
function IKChord(DeviceValue: THandle; var PrintRect: TRect; x1, y1, x2, y2: Integer; var DrawInfo: IKPRINT_DRAWINFO; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
PrintRect	境界矩形を指定する構造体変数
x1,y1	最初の径の端点の x,y 座標
x2,y2	次の径の端点の x,y 座標
DrawInfo	描画する情報を指定する構造体変数
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PrintRect で指定した矩形に含まれる楕円弧を(x1,y1),(x2,y2)の間を通る線によって 2 等分された領域を描画します。アウトラインは **PenStyle** の値で描画され、内部は **BrushStyle** の値で塗りつぶされます。
(対象はスクリーン、プリンタ、メモリハンドル)
描画するには DrawInfo の **PenWidth**,**PenStyle**,**PenMode**,**PenColor**,**BrushStyle**,**BrushColor**,**Transparent**,**BackColor** に値を設定する必要があります。**BackColor** は **Transparent** = 0 でペンが実線以外、あるいはブラシがハッチパターンの場合に有効です。

PrintRect のメンバー変数と x1,y1,x2,y2 の値をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
PrintRect のメンバー変数と x1,y1,x2,y2 の値を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5Chord:	DeviceValue, PrintRect, x1, y1, x2, y2, PrintInfo
IKChord:	DeviceValue, PrintRect, x1, y1, x2, y2, DrawInfo, DeviceMode

PrintInfo と DrawInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。Transparent を 0 以外に設定し、IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

【ImageKit6 との違い】
IKPRINT_DRAWINFO 構造体に BackColor が追加されました。Transparent が 0 以外であれば IK6 と同じ動作です。

IKDrawFocusRect

【機能】
画面にフォーカスを示す矩形を描画します。

【関数書式】
(1)C++Builder
BOOL IKDrawFocusRect(HDC hDC, LPRECT PrintRect, PTR_IKPRINT_DRAWINFO DrawInfo, BYTE UnitMode);
(2)Delphi
function IKDrawFocusRect(hDC: HDC; var PrintRect: Trect; var DrawInfo: IKPRINT_DRAWINFO; UnitMode: Byte): LongBool;

【引数】

名称	内容
hDC	デバイスコンテキスト
PrintRect	境界矩形を指定する構造体変数
DrawInfo	描画する情報を指定する構造体変数
UnitMode	PrintRect の単位(0:ピクセル単位、1:0.1mm 単位)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
XOR 機能を使用して矩形を描画するため、同じ矩形を引数にして 2 回呼び出すと画面から矩形が消去されます。描画した矩形を含む領域をスクロールするには、再度矩形を描画して画面から消去し、領域をスクロールした後にもう一度新しい位置に同じ矩形を描画します。
描画するには DrawInfo の **BrushStyle**,**BrushColor** に値を設定する必要があります。

PrintRect のメンバー変数をピクセル単位として扱う場合
UnitMode が 0
PrintRect のメンバー変数を 0.1mm 単位として扱う場合
UnitMode が 1

IKDrawString

【機能】
GDI+の機能を利用して描画先オブジェクトにテキストを描画します。

【関数書式】
(1)C++Builder
BOOL IKDrawString(LPVOID DeviceValue, LPRECT PrintRect, short BrushType, short Style, HANDLE imgHandle, short TextRenderingHint, int FormatFlags, PTR_IKPRINT_TEXTINFO TextInfo, LPCTSTR Text, BYTE DeviceMode);
(2)Delphi
function IKDrawString(DeviceValue: THandle; var PrintRect: TRect; BrushType, Style: Smallint; imgHandle: THandle; TextRenderingHint: Smallint; FormatFlags: Integer; var TextInfo: IKPRINT_TEXTINFO; Text: PChar; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
PrintRect	描画開始位置あるいは描画領域を指定する構造体
BrushType	ブラシの種類(0:ソリッド、1:ハッチ、2:テクスチャ、4:グラデーション)
Style	ブラシ毎のスタイル(BrushType が 0 以外の場合に有効)
imgHandle	テクスチャブラシ(BrushType=4)の時に使用するラスタイメージのメモリハンドル
TextRenderingHint	テキストのレンダリングモード(デフォルトは 0)
FormatFlags	文字列の書式情報(デフォルトは 0)
TextInfo	描画情報を指定する構造体
Text	描画する文字列(終端はヌル)
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PrintRect の right あるいは bottom を 0 に設定すると left と top を始点としてテキストを描画します。それ以外の場合は PrintRect で囲まれる矩形の内部にテキストが描画され、内部に収まらないテキストは切り捨てられます。(対象はスクリーン、プリンタ、メモリハンドル)
描画するには TextInfo の **TextColor1,Alpha1,TextColor2,Alpha2,FontName,FontSize,CharAngle,HCentering, VCentering,HotkeyPrefix** に値を設定する必要があります。**HotkeyPrefix** に 1 以外を設定するとプリフィックス文字の処理を行いません(&A は A にはならず、&A もしくは A と描画されます)。
TextColor1,Alpha1 は「ソリッドブラシ:ブラシの色」「ハッチブラシ:描画される線の色」「グラデーションブラシ:線形グラデーションの開始色」を表します。
TextColor2,Alpha2 は「ハッチブラシ:線間の領域の色」「グラデーションブラシ:線形グラデーションの終了色」を表します。
グラデーションブラシ(BrushType=4)を使用する場合は、PrinRect に矩形領域を設定してください。
当関数を実行する場合は、事前に IKPrintGdipStart 関数を実行する必要があります。

PrintRect のメンバー変数をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
PrintRect のメンバー変数を 0.1mm 単位として扱う場合
DeviceMode が 1

「ブラシ毎のスタイル(Style に設定する値)」
ハッチブラシの場合:

値	説明
0	水平線のパターン。
1	垂直線のパターン。
2	左上から右下への対角線のパターン。
3	右上から左下への対角線のパターン。

4	交差する水平および垂直の線を指定します。
5	交差する右上がりと左上がりの対角線を指定します。線はアンチエイリアス処理されます。
6	5%のハッチを指定します。前景色の背景色に対する割合は、5: 100 です。
7	10%のハッチを指定します。前景色の背景色に対する割合は、10: 100 です。
8	20%のハッチを指定します。前景色の背景色に対する割合は、20: 100 です。
9	25%のハッチを指定します。前景色の背景色に対する割合は、25: 100 です。
10	30%のハッチを指定します。前景色の背景色に対する割合は、30: 100 です。
11	40%のハッチを指定します。前景色の背景色に対する割合は、40: 100 です。
12	50%のハッチを指定します。前景色の背景色に対する割合は、50: 100 です。
13	60%のハッチを指定します。前景色の背景色に対する割合は、60: 100 です。
14	70%のハッチを指定します。前景色の背景色に対する割合は、70: 100 です。
15	75%のハッチを指定します。前景色の背景色に対する割合は、75: 100 です。
16	80%のハッチを指定します。前景色の背景色に対する割合は、80: 100 です。
17	90%のハッチを指定します。前景色の背景色に対する割合は、90: 100 です。
18	上の点から下の点へ右に傾斜し、2 よりも間隔が 50%狭く、アンチエイリアス処理されない対角線。
19	上の点から下の点へ左に傾斜し、3 よりも間隔が 50%狭く、アンチエイリアス処理されない対角線。
20	上の点から下の点へ右に傾斜し、2 よりも間隔が 50%狭く、幅が 2 倍の対角線。このハッチパターンはアンチエイリアス処理されません。
21	上の点から下の点へ左に傾斜し、3 よりも間隔が 50%狭く、幅が 2 倍で、線がアンチエイリアス処理されない対角線。
22	上の点から下の点へ右に傾斜し、2 と間隔が等しく、幅が 3 倍で、アンチエイリアス処理されない対角線。
23	上の点から下の点へ左に傾斜し、3 と間隔が等しく、幅が 3 倍で、アンチエイリアス処理されない対角線。
24	間隔が 1 よりも 50%狭い垂直線を指定します。
25	間隔が 0 よりも 50%狭い水平線を指定します。
26	間隔が 1 よりも 75%狭い(または 24 よりも 25%狭い)垂直線を指定します。
27	間隔が 0 よりも 75%狭い(または 25 よりも 25%狭い)水平線を指定します。
28	1 よりも間隔が 50%狭く、幅が 2 倍の垂直線を指定します。
29	0 よりも間隔が 50%狭く、幅が 2 倍の水平線を指定します。
30	上の点から下の点へ右に傾斜した、破線の対角線を指定します。
31	上の点から下の点へ左に傾斜した、破線の対角線を指定します。
32	破線の水平線を指定します。
33	破線の垂直線を指定します。
34	紙吹雪のように見えるハッチを指定します。
35	34 よりも大きいピースから構成される、紙吹雪のように見えるハッチを指定します。
36	ジグザグに構成された水平線を指定します。
37	ティルダで構成された水平線を指定します。
38	上の点から下の点へ左に傾斜した、積み重ねたレンガ状のハッチを指定します。
39	レンガを水平に積み上げたように見えるハッチを指定します。
40	織物のように見えるハッチを指定します。
41	格子柄の生地のように見えるハッチを指定します。
42	芝生のように見えるハッチを指定します。
43	それぞれがドットで構成されて交差する、水平線と垂直線を指定します。
44	それぞれがドットで構成されて交差する、右上がりと左上がりの対角線を指定します。
45	上の点から下の点へ右に傾斜した、対角線状に積み重ねた板屋根のように見えるハッチを指定します。
46	四目格子のように見えるハッチを指定します。
47	交互に並べた球体のように見えるハッチを指定します。
48	間隔が 4 よりも 50%狭い、交差する水平線と垂直線を指定します。
49	チェッカーボードのように見えるハッチを指定します。
50	49 の 2 倍のサイズの正方形による、チェッカーボードのように見えるハッチを指定します。
51	交差する右上がりと左上がりの線で、アンチエイリアス処理されない対角線を指定します。
52	斜めに置かれたチェッカーボードのように見えるハッチを指定します。

テクスチャブラシの場合:

値	説明
0	グラデーションまたはテクスチャを並べて表示します。
1	テクスチャまたはグラデーションを水平方向に反転し、それを並べて表示します。
2	テクスチャまたはグラデーションを垂直方向に反転し、それを並べて表示します。
3	テクスチャまたはグラデーションを水平および垂直方向に反転し、それを並べて表示します。

4	テクスチャまたはグラデーションをオブジェクトの端に揃えます。

グラデーションブラシの場合:

値	説明
0	左から右へのグラデーションを指定します。
1	上から下へのグラデーションを指定します。
2	左上から右下へのグラデーションを指定します。
3	右上から左下へのグラデーションを指定します。

「テキストのレンダリングモード」

値	説明
0	グリフビットマップを使用し、システム既定のレンダリングヒントで各文字を描画することを指定します。
1	グリフビットマップを使用して、各文字を描画することを指定します。ヒンティングを使用して、文字のステム部分と曲線部分の見た目を向上します。
2	グリフビットマップを使用して、各文字を描画することを指定します。ヒンティングは使用されません。
3	アンチエイリアス処理されたグリフビットマップを使用して、ヒンティングありで各文字を描画することを指定します。アンチエイリアスによってより高い品質が得られますが、パフォーマンスは大きく低下します。
4	アンチエイリアス処理されたグリフビットマップを使用して、ヒンティングなしに各文字を描画することを指定します。アンチエイリアスによって品質が向上します。ヒンティングがオフにされるため、ステム幅の違いが目立ちます。
5	グリフ CT ビットマップを使用して、ヒンティングありで各文字を描画することを指定します。最高の品質設定です。ClearType テキストフォント機能を利用するときに使用します。

「文字列の書式情報」
設定値は次の値の論理和となります。

値	説明
0x00000001	テキストの方向を右から左に指定します。
0x00000002	テキストが縦書きになるよう指定します。
0x00000004	グリフが外接する四角形にかからないよう指定します。既定では、端に表示する必要がある場合、一部のグリフが若干四角形にかかるよう設定されます。
0x00000020	左から右を指示するマークなどの制御文字をグリフで表現します。
0x00000400	フォールバックを無効にして、要求されたフォントでサポートされていない文字のフォントを切り替えます。欠落文字は、グリフの欠落したフォント(通常は空白の正方形)で表示されます。
0x00000800	既定では、各行末の空白が除外されます。各行末の空白を計測に含める場合はこの値を設定します。
0x00001000	四角形内の書式指定時に、行間のテキストのラップを無効にします。この値は、四角形ではなく点が渡された場合、または長さゼロの行の四角形が指定された場合に暗黙的に指定されます。
0x00002000	書式指定用の四角形には、完全な直線だけがレイアウトされます。既定では、クリッピングの結果、テキストの末尾が表示された状態、または行が表示されなくなった状態のうち、いずれか早い方の状態になるまでレイアウトが継続します。既定の設定では、行高さの整数倍でない書式指定用四角形を用いた場合は、なるまでレイアウトが継続します。既定の設定では、行高さの整数倍でない書式指定用四角形を用いた場合は、最後の行の一部が隠れることがあります。必ず行全体が表示されるようにするには、この値を指定した上で、少なくとも 1 つの行と高さが同じの書式指定用の四角形をご使用ください。
0x00004000	論理グリフの突出部と書式指定用の四角形からはみ出すラップされていないテキストを表示できます。既定では、書式指定用の四角形からはみ出たテキストとグリフ部はすべてクリップされます。

※16 進表記のため、Delphi は 0x を$に置き換えてください。

コード例:
(1)C++Builder

```
IKPRINT_TEXTINFO TextInfo;
float Str_Width, Str_Height;
int charactersFitted, linesFilled;
RECT PrintRect;
ULONG_PTR token;//DWORD token;
```

```
token = IKPrintGdipStart();
if (token == 0) return;

memset(&TextInfo, 0, sizeof(IKPRINT_TEXTINFO));
TextInfo.FontSize = 20;
lstrcpy(TextInfo.FontName, "MS ゴシック");
TextInfo.HotkeyPrefix = 0;
IKMeasureString(hDC, 0, 0, &Str_Width, &Str_Height, &charactersFitted, &linesFilled, 0, &TextInfo, "テキスト", 0);

PrintRect.left = 10;
PrintRect.top = 10;
PrintRect.right = PrintRect.left + (long)Str_Width - 1;
PrintRect.bottom = PrintRect.top + (long)Str_Height - 1;
TextInfo.CharAngle = 0;
TextInfo.TextColor1 = RGB(255, 0, 0);
TextInfo.Alpha1 = 255;
TextInfo.TextColor2 = RGB(255, 255, 255);
TextInfo.Alpha2 = 255;
//グラデーションブラシを使用
IKDrawString(hDC, &PrintRect, 4, 0, 0, 0, 0, &TextInfo, "テキスト", 0);
IKPrintGdipEnd(token)
```

(2)Delphi

```
TextInfo: IKPRINT_TEXTINFO;
Str_Width, Str_Height: Single;
CharactersFitted, linesFilled: Integer;
PrintRect: TRect;
token: DWORD;

token := IKPrintGdipStart();
if token = 0 then Exit;

FillChar(TextInfo, SizeOf(TextInfo), 0);
TextInfo.FontSize := 20;
StrPCopy(TextInfo.FontName, 'MS ゴシック');
TextInfo.HotkeyPrefix := 0;
IKMeasureString(DC, 0, 0, Str_Width, Str_Height, charactersFitted, linesFilled, 0, TextInfo, 'テキスト', 0);

PrintRect.Left := 10;
PrintRect.Top := 10;
PrintRect.Right := PrintRect.Left + Trunc(Str_Width) - 1;
PrintRect.Bottom := PrintRect.Top + Trunc(Str_Height) - 1;
TextInfo.CharAngle := 0;
TextInfo.TextColor1 := clRed;
TextInfo.Alpha1 := 255;
TextInfo.TextColor2 := clWhite;
TextInfo.Alpha2 := 255;
//グラデーションブラシを使用
IKDrawString(DC, PrintRect, 4, 0, 0, 0, 0, TextInfo, 'テキスト', 0);
IKPrintGdipEnd(token)
```

IKDrawText

【機能】
描画先オブジェクトにテキストを描画します。

【関数書式】
(1)C++Builder
BOOL IKDrawText(LPVOID DeviceValue, LPRECT PrintRect, PTR_IKPRINT_TEXTINFO TextInfo, LPCTSTR Text, BOOL EnableFontScale, BOOL Clip, BYTE DeviceMode);
(2)Delphi
function IKDrawText(DeviceValue: THandle; var PrintRect: TRect; var TextInfo: IKPRINT_TEXTINFO; Text: PChar; EnableFontScale, Clip: LongBool; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
PrintRect	始点(left,top)、終点(right,bottom)を指定する構造体変数
TextInfo	描画する情報を指定する構造体変数
Text	描画する文字列(終端はヌル)
EnableFontScale	フォントのスケーリング [True(0 以外):する、False(0):しない]
Clip	クリッピング [True(0 以外):する、False(0):しない]
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PrintRect で指定した座標を矩形とし、その中にテキストを描画します。(対象はスクリーン、プリンタ、メモリハンドル)
EnableFontScale が True(0 以外)の場合は、文字列が矩形内に収まりきらない場合にフォントをスケーリングします。ただし、スケーリングを行っても文字列が矩形内に収まりきらない場合は、Clip によって状態が変わります。(例えば、一番小さいフォントサイズを使用しても文字列が矩形内に収まりきらない場合など。)
EnableFontScale が False(0)の場合は、指定されたフォントをそのまま使用し、文字列が境界矩形に収まりきらない場合は収まる最後の単語の切れ目で折り返し、CRLF でも折り返しを行います。(複数行の描画可)
Clip が True(0 以外)の場合は、矩形内に収まりきらない文字列をカットしますが、False(0)の場合は収まりきらない文字列も描画します。矩形内に文字列が収まる場合は、Clip は意味を持ちません。文字列の描画方向は、文字列の方向と文字の回転に関わらず、与えられた開始位置から右もしくは下方向に描画します。
描画するには TextInfo の **CharSet,TextColor1,TextColor2,FontName,FontSize,Transparent,Direction,CharAngle,CharExtra,HCentering,VCentering,HotkeyPrefix** に値を設定する必要があります。**CharAngle** は 0,90,180,270 のみ有効です。EnableFontScale が False(0)の場合は **CharAngle,Direction,VCentering** の設定は無効です。**HotkeyPrefix** を 0 に設定するとプリフィックス文字の処理を行いません(&A は A にはならず、&A と描画されます)。

PrintRect のメンバー変数をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
PrintRect のメンバー変数を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5DrawText:	DeviceValue, PrintRect, PrintInfo
IKDrawText:	DeviceValue, PrintRect, TextInfo, Text, EnableFontScale, Clip, DeviceMode

PrintInfo と TextInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。
IK5PRINT_DATA --> IKPRINT_TEXTINFO
CharExtra,CharSet,RotateString,Alpha1,Alpha2,HotkeyPrefix が新たに追加され、Text が構造体から削除されました。
TextForeColor,TextBackColor が TextColor1,TextColor2 にそれぞれ変更されました。
PrintInfo.Text を引数の Text、IK5SetDeviceMode で設定した値を引数の DeviceMode、TextInfo の CharSet を-1、CharExtra を 0、HotkeyPrefix を 1 に設定し、TextForeColor,TextBackColor を TextColor1,TextColor2 にそれぞれ変更す

ると IK5 と同じ動作となります。

【ImageKit6 との違い】

IKPRINT_TEXTINFO 構造体に Alpha1,Alpha2,HotkeyPrefix が追加され、TextForeColor,TextBackColor が TextColor1,TextColor2 にそれぞれ変更されました。HotkeyPrefix を 1 に設定し、TextForeColor,TextBackColor を TextColor1,TextColor2 にそれぞれ変更すると IK6 と同じ動作となります。

IKEllipse

【機能】
描画先オブジェクトに楕円を描画します。

【関数書式】
(1)C++Builder
BOOL IKEllipse(LPVOID DeviceValue, LPRECT PrintRect, PTR_IKPRINT_DRAWINFO DrawInfo, BYTE DeviceMode);
(2)Delphi
function IKEllipse(DeviceValue: THandle; var PrintRect: TRect; var DrawInfo: IKPRINT_DRAWINFO; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
PrintRect	矩形を指定する構造体変数
DrawInfo	描画する情報を指定する構造体変数
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PrintRect で指定した矩形に内接する円を描画します。アウトラインは **PenStyle** の値で描画され、内部は **BrushStyle** の値で塗りつぶされます。(対象はスクリーン、プリンタ、メモリハンドル)
描画するには DrawInfo の **PenWidth**,**PenStyle**,**PenMode**,**PenColor**,**BrushStyle**,**BrushColor**,**Transparent**,**BackColor** に値を設定する必要があります。**BackColor** は **Transparent** = 0 でペンが実線以外、あるいはブラシがハッチパターンの場合に有効です。

PrintRect のメンバー変数をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
PrintRect のメンバー変数を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5Ellipse:	DeviceValue, PrintRect, PrintInfo
IKEllipse:	DeviceValue, PrintRect, DrawInfo, DeviceMode

PrintInfo と DrawInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。Transparent を 0 以外に設定し、IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

【ImageKit6 との違い】
IKPRINT_DRAWINFO 構造体に BackColor が追加されました。Transparent が 0 以外であれば IK6 と同じ動作です。

IKEnumPaperBins

【機能】
指定されたプリンタ名からサポートされている用紙トレイの名前、トレイの番号を取得します。

【関数書式】
(1)C++Builder
int IKEnumPaperBins(LPCTSTR PrinterName, LPTSTR BinNames, LPWORD BinNumbers);
(2)Delphi
function IKEnumPaperBins(PrinterName, BinNames: PChar; var BinNumbers: Word): Integer;

【引数】

名称	内容
PrinterName	プリンタ名
BinNames	用紙トレイの名前を取得する文字列 NULL(C++Builder),nil(Delphi)を設定した場合は、戻り値として配列に必要な要素数を返します。
BinNumbers	用紙トレイの番号を取得する配列 (1)C++Builder　配列の先頭のポインタを渡す (2)Delphi　Bnum: array [0..2] of Word の場合は、Bnum[0]を与える

【戻り値】
取得した項目数、あるいは配列に必要な要素数を返します。(0 は失敗)

【解説】
BinNames にNULL(C++Builder),nil(Delphi)を設定した場合の戻り値はそのままBinNumbers の配列の要素数になりますが、BinNames はその要素数×(24＋1)＋1 の領域を割り当てなければなりません。配列に必要な要素数はIKPrintGetArrayNum 関数でも取得可能です。
BinNames 文字列は"xxxxx,xxxxx,xxxxx,・・・・・・・・,"となり、名称はカンマで区切られて設定されます(終了は","です)。

コード例:
(1)C++Builder
```
int Size;
TCHAR *BinNames;
LPWORD BinNumbers;

Size = IKEnumPaperBins("EPSON LP-8200C", NULL, NULL);
if (Size < 1) return;
BinNames = (TCHAR *)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, Size * (sizeof(TCHAR) * (24 + 1)) +
sizeof(TCHAR));
BinNumbers = (LPWORD)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, sizeof(WORD) * Size);
_try {
  IKEnumPaperBins("EPSON LP-8200C", BinNames, BinNumbers);
  // 様々な処理
  ・・・・・・・
} _finally {
  HeapFree(GetProcessHeap(), 0, BinNames);
  HeapFree(GetProcessHeap(), 0, BinNumbers);
}
```
(2)Delphi
```
Size: Integer;
BinNames: PChar;
BinNumbers: array of Word;

Size := IKEnumPaperBins('EPSON LP-8200C', nil, PWord(nil)^);
if Size < 1 then Exit;
```

```
GetMem(BinNames, Size * (SizeOf(Char) * (24 + 1)) + SizeOf(Char));
SetLength(BinNumbers, Size);
try
  IKEnumPaperBins('EPSON LP-8200C', BinNames, BinNumbers[0]);
  // 様々な処理
  ・・・・・・・
finally
  FreeMem(BinNames);
end;
```

IKEnumPaperSizes

【機能】
指定されたプリンタ名からサポートされている用紙の名前、サイズ番号、寸法を取得します。

【関数書式】
(1)C++Builder
int IKEnumPaperSizes(LPCTSTR PrinterName, LPTSTR PaperNames, LPWORD PaperNumbers, LPPOINT PaperSizes);
(2)Delphi
function IKEnumPaperSizes(PrinterName, PaperNames: PChar; var PaperNumbers: Word; var PaperSizes: TPoint): Integer;

【引数】

名称	内容
PrinterName	プリンタ名
PaperNames	用紙の名前を取得する文字列 NULL(C++Builder),nil(Delphi)を設定した場合は、戻り値として配列に必要な要素数を返します。
PaperNumbers	用紙のサイズ番号を取得する配列 (1)C++Builder　配列の先頭のポインタを渡す (2)Delphi　Pnum: array [0..2] of Word の場合は、Pnum[0]を与える
PaperSizes	用紙の寸法を取得する配列 (1)C++Builder　配列の先頭のポインタを渡す (2)Delphi　Ps: array [0..2] of TPoint の場合は、Ps[0]を与える

【戻り値】
取得した項目数、あるいは配列に必要な要素数を返します。(0 は失敗)

【解説】
PaperNames に NULL(C++Builder),nil(Delphi)を設定した場合の戻り値はそのまま PaperNumbers と PaperSizes の配列の要素数になりますが、PaperNames はその要素数×(64＋1)＋1 の領域を割り当てなければなりません。配列に必要な要素数は IKPrintGetArrayNum 関数でも取得可能です。
PaperNames 文字列は"xxxxx,xxxxx,xxxxx,・・・・・・・・,"となり、名称はカンマで区切られて設定されます(終了は","です)。
PaperSizes は各用紙サイズの寸法を 0.1mm 単位で取得します。印刷の向きを縦方向として **x** に用紙の幅、**y** に用紙の長さを格納します。

コード例:
(1)C++Builder

```
int Size;
TCHAR *PaperNames;
LPWORD PaperNumbers;
LPPOINT PaperSizes;

Size = IKEnumPaperSizes("EPSON LP-8200C", NULL, NULL, NULL);
if (Size < 1) return;
PaperNames = (TCHAR *)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, Size * (sizeof(TCHAR) * (64 + 1)) + sizeof(TCHAR));
PaperNumbers = (LPWORD)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, sizeof(WORD) * Size);
PaperSizes = (LPPOINT)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, sizeof(POINT) * Size);
__try {
  IKEnumPaperSizes("EPSON LP-8200C", PaperNames, PaperNumbers, PaperSizes);
  // 様々な処理
  ・・・・・・・
} __finally {
```

```
    HeapFree(GetProcessHeap(), 0, PaperNames);
    HeapFree(GetProcessHeap(), 0, PaperNumbers);
    HeapFree(GetProcessHeap(), 0, PaperSizes);
  }
```

(2)Delphi

```
  Size: Integer;
  PaperNames: PChar;
  PaperNumbers: array of Word;
  PaperSizes: array of TPoint;

  Size := IKEnumPaperSizes('EPSON LP-8200C', nil, PWord(nil)^, PPoint(nil)^);
  if Size < 1 then Exit;
  GetMem(PaperNames, Size * (SizeOf(Char) * (64 + 1)) + SizeOf(Char));
  SetLength(PaperNumbers, Size);
  SetLength(PaperSizes, Size);
  try
    IKEnumPaperSizes('EPSON LP-8200C', PaperNames, PaperNumbers[0], PaperSizes[0]);
    // 様々な処理
    .......
  finally
    FreeMem(PaperNames);
  end;
```

IKEnumPorts

【機能】
ポートを列挙します。

【関数書式】
(1)C++Builder
DWORD IKEnumPorts(LPTSTR Ports);
(2)Delphi
function IKEnumPorts(Ports: PChar): DWORD;

【引数】

名称	内容
Ports	ポート名を取得する文字列

【戻り値】
Ports に NULL(C++Builder),nil(Delphi)を設定した場合は、ポート名を取得する際に必要な文字列のサイズを返します(終端のヌルは除く)。Ansi 版はバイト数で Unicode 版は文字数となります。
Ports に文字列を設定した場合は、取得したポートの数を返します。

【解説】
※ポート名を取得する場合はサイズを指定して確保してください。
(1)C++Builder
TCHAR Ports[1024];
(2)Delphi
Ports: array[0..1023] of Char;

Ports 文字列は"xxxxx,xxxxx,xxxxx,・・・・・・・・,"となり、名称はカンマで区切られて設定されます(終了は","です)。

IKEnumPrinters

【機能】
インストールされているプリンタを列挙します。

【関数書式】
(1)C++Builder
DWORD IKEnumPrinters(LPTSTR Printers, short *DefaultPrinterNo);
(2)Delphi
function IKEnumPrinters(Printers: PChar; var DefaultPrinterNo: Smallint): DWORD;

【引数】

名称	内容
Printers	プリンタ名を取得する文字列
DefaultPrinterNo	通常使うプリンタ(デフォルトプリンタ)の番号(0～)

【戻り値】
Printers に NULL(C++Builder),nil(Delphi)を設定した場合は、プリンタ名を取得する際に必要な文字列のサイズを返します(終端のヌルは除く)。Ansi 版はバイト数で Unicode 版は文字数となります。
Printers に文字列を設定した場合は、取得したプリンタの数を返します。

【解説】
※プリンタ名を取得する場合はサイズを指定して確保してください。
(1)C++Builder
TCHAR Printers[1024];
(2)Delphi
Printers: array[0..1023] of Char;

Printers 文字列は"xxxxx,xxxxx,xxxxx,・・・・・・・・,"となり、名称はカンマで区切られて設定されます(終了は","です)。
DefaultPrinterNo は Printers 文字列に設定されたプリンタの中で、通常使うプリンタの番号が何番かを示します。最初のプリンタを示す場合は 0 となります。

IKEnumResolutions

【機能】
指定されたプリンタ名からサポートされている解像度のリストを取得します。

【関数書式】
(1)C++Builder
```
int IKEnumResolutions(LPCTSTR PrinterName, LPPOINT Resolutions);
```
(2)Delphi
```
function IKEnumResolutions(PrinterName: PChar; var Resolutions: TPoint): Integer;
```

【引数】

名称	内容
PrinterName	プリンタ名
Resolutions	解像度のリストを取得する配列 NULL(C++Builder),nil(Delphi)を設定した場合は、戻り値として配列に必要な要素数を返します。 (1)C++Builder　配列の先頭のポインタを渡す (2)Delphi　Res: array [0..2] of Word の場合は、Res[0]を与える

【戻り値】
取得した項目数、あるいは配列に必要な要素数を返します。(0 は失敗)

【解説】
Resolutions に NULL(C++Builder),nil(Delphi)を設定した場合の戻り値はそのまま配列の要素数になります。配列に必要な要素数は IKPrintGetArrayNum 関数でも取得可能です。
Resolutions には 1 インチ当たりのドット数(dpi)単位で水平解像度と垂直解像度を示す値が格納されます。**x** が水平解像度、**y** が垂直解像度になります。

コード例:
(1)C++Builder
```
int Size;
LPPOINT Resolutions;

Size = IKEnumResolutions("EPSON LP-8200C", NULL);
if (Size < 1) return;
Resolutions = (LPPOINT)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, sizeof(POINT) * Size);
_try {
  IKEnumResolutions("EPSON LP-8200C", Resolutions);
  // 様々な処理
  ・・・・・・・
} _finally {
  HeapFree(GetProcessHeap(), 0, Resolutions);
}
```
(2)Delphi
```
Size: Integer;
Resolutions: array of TPoint;

Size := IKEnumResolutions('EPSON LP-8200C', PPoint(nil)^);
if Size < 1 then Exit;
SetLength(Resolutions, Size);
IKEnumResolutions('EPSON LP-8200C', Resolutions[0]);
// 様々な処理
・・・・・・・
```

IKFillRect

【機能】
描画先オブジェクトに対して指定した矩形を塗りつぶします。

【関数書式】
(1)C++Builder
BOOL IKFillRect(LPVOID DeviceValue, LPRECT PrintRect, PTR_IKPRINT_DRAWINFO DrawInfo, BYTE DeviceMode);
(2)Delphi
function IKFillRect(DeviceValue: THandle; var PrintRect: TRect; var DrawInfo: IKPRINT_DRAWINFO; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
PrintRect	矩形を指定する構造体変数
DrawInfo	描画する情報を指定する構造体変数
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PrintRect で指定した矩形を **BrushStyle** の値で塗りつぶします。ただし、矩形の左と上の端は領域に含まれますが、右と下の端は含まれません。(対象はスクリーン、プリンタ、メモリハンドル)
描画するには DrawInfo の **BrushStyle,BrushColor,Transparent,BackColor** に値を設定する必要があります。**BackColor** は **Transparent** = 0 でハッチパターンブラシの場合に有効です。**Transparent** が 0 以外でハッチパターンブラシを選択した場合、線と線の間は透過されません(既定の色で塗りつぶされます)。線と線の間を透過する場合は IKRectangle などをご使用ください。

PrintRect のメンバー変数をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
PrintRect のメンバー変数を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5FillRect:	DeviceValue, PrintRect, PrintInfo
IKFillRect:	DeviceValue, PrintRect, DrawInfo, DeviceMode

PrintInfo と DrawInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。Transparent を 0 以外に設定し、IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

【ImageKit6 との違い】
IKPRINT_DRAWINFO 構造体に BackColor が追加されました。Transparent が 0 以外であれば IK6 と同じ動作です。

IKFrameRect

【機能】
描画先オブジェクトに対して指定した矩形の境界を描画します。

【関数書式】
(1)C++Builder
BOOL IKFrameRect(LPVOID DeviceValue, LPRECT PrintRect, PTR_IKPRINT_DRAWINFO DrawInfo, BYTE DeviceMode);
(2)Delphi
function IKFrameRect(DeviceValue: THandle; var PrintRect: TRect; var DrawInfo: IKPRINT_DRAWINFO; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
PrintRect	矩形を指定する構造体変数
DrawInfo	描画する情報を指定する構造体変数
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PrintRect で指定した矩形の境界を **BrushStyle** の値で描画します。境界の内部は塗りつぶされません。(対象はスクリーン、プリンタ、メモリハンドル)
描画するには DrawInfo の **BrushStyle,BrushColor,Transparent,BackColor** に値を設定する必要があります。**BackColor** は **Transparent** = 0 でハッチパターンブラシの場合に有効です。**Transparent** が 0 以外でハッチパターンブラシを選択した場合、線と線の間は透過されません(既定の色で塗りつぶされます)。線と線の間を透過する場合は IKRectangle などをご使用ください。

PrintRect のメンバー変数をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
PrintRect のメンバー変数を 0.1mm 単位として扱う場合
DeviceMode が 1

IKGetDefaultPrinter

【機能】
通常使うプリンタ(デフォルトプリンタ)を取得します。

【関数書式】
(1)C++Builder
DWORD IKGetDefaultPrinter(LPTSTR PrinterName);
(2)Delphi
function IKGetDefaultPrinter(PrinterName: PChar): DWORD;

【引数】

名称	内容
PrinterName	プリンタ名を取得する変数

【戻り値】
PrinterName に設定された文字列の長さを返します(終端のヌルは除く、実行に失敗した場合は 0)。

【解説】
Windows のプリンタフォルダに登録されている通常使うプリンタを取得します。
IKEnumPrinters 関数でも通常使うプリンタを取得できます。

IKGetDevModeHandle

【機能】
指定したプリンタの DEVMODE 構造体へのポインタのハンドルを取得します。

【関数書式】
(1)C++Builder
HANDLE IKGetDevModeHandle(LPCTSTR PrinterName, LPCTSTR PrintFileName);
(2)Delphi
function IKGetDevModeHandle(PrinterName, PrintFileName: PChar): THandle;

【引数】

名称	内容
PrinterName	プリンタ名
PrintFileName	プリンタ設定ファイル名（**IKSaveDevModeHandle(Ex)**もしくは **IKSetPrint** で保存したファイル名）

【戻り値】
DEVMODE 構造体のハンドル（実行に失敗した場合はヌル(0)が返されます。）

【解説】
PrinterName にはプリンタ名を、PrintFileName には **IKSaveDevModeHandle(Ex)**もしくは **IKSetPrint** で保存したプリンタ設定ファイル名を設定してください。両方設定した場合は PrinterName が有効になります。どちらか片方を設定する場合は必要ない引数に空文字列を設定してください。(″″,'',NULL,nil)
取得した DEVMODE 構造体のハンドルは、IKPrintCreateDC(Ex)などで使用します。DEVMODE 構造体のハンドルを解放する場合は、IKReleaseDevModeHandle(Ex)を実行します。DEVMODE 構造体については WindowsAPI 関連の書籍などをご覧ください。

IKGetDevModeHandleEx

【機能】
指定したプリンタの DEVMODE 構造体と DEVNAMES 構造体へのポインタのハンドルを取得します。

【関数書式】
(1)C++Builder
HANDLE IKGetDevModeHandleEx(LPCTSTR PrinterName, LPCTSTR PrintFileName, LPHANDLE hDevNames);
(2)Delphi
function IKGetDevModeHandleEx(PrinterName, PrintFileName: PChar; hDevNames: PHandle): THandle;

【引数】

名称	内容
PrinterName	プリンタ名
PrintFileName	プリンタ設定ファイル名（**IKSaveDevModeHandle(Ex)**もしくは **IKSetPrint** で保存したファイル名）
hDevNames	取得する DEVNAMES 構造体のハンドル

【戻り値】
DEVMODE 構造体のハンドル（実行に失敗した場合はヌル(0)が返されます。）

【解説】
PrinterName にはプリンタ名を、PrintFileName には **IKSaveDevModeHandle(Ex)**もしくは **IKSetPrint** で保存したプリンタ設定ファイル名を設定してください。両方設定した場合は PrinterName が有効になります。どちらか片方を設定する場合は必要ない引数に空文字列を設定してください。(″″,'',NULL,nil)
取得した DEVMODE 構造体と DEVNAMES 構造体のハンドルは、IKPrintCreateDCEx や IKPrintDialog で使用します。
DEVMODE 構造体と DEVNAMES 構造体のハンドルを解放する場合は、IKReleaseDevModeHandleEx を実行します。
DEVMODE 構造体と DEVNAMES 構造体については WindowsAPI 関連の書籍などをご覧ください。

IKGetDevModeInfo

【機能】
DEVMODE 構造体のハンドルから印刷情報を取得します。

【関数書式】
(1)C++Builder
BOOL IKGetDevModeInfo(HANDLE hDevMode, PTR_IKPRINT_DEVMODEINFO DevModeInfo);
(2)Delphi
function IKGetDevModeInfo(hDevMode: THandle; var DevModeInfo: IKPRINT_DEVMODEINFO): LongBool;

【引数】

名称	内容
hDevMode	**IKGetDevModeHandle(Ex)**で取得した DEVMODE 構造体のハンドル
DevModeInfo	印刷情報を取得する構造体(ユーザ定義型)変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
成功した場合、DevModeInfo に印刷情報が設定されます。
DEVMODE 構造体については WindowsAPI 関連の書籍などをご覧ください。
IKPRINT_DEVMODEINFO については、**「Ik10Print.dll/Ik10PrintA.dll/Ik10Print64.dll/Ik10Print64A.dll」の構造体の定義**をご覧ください。

IKGetDevNamesInfo

【機能】
DEVNAMES 構造体のハンドルから文字列情報を取得します。

【関数書式】
(1)C++Builder
BOOL IKGetDevNamesInfo(HANDLE hDevNames, LPTSTR DriverName, LPTSTR DeviceName, LPTSTR OutputName);
(2)Delphi
function IKGetDevNamesInfo(hDevNames: THandle; DriverName, DeviceName, OutputName: PChar): LongBool;

【引数】

名称	内容
hDevNames	**IKGetDevModeHandleEx** で取得した DEVNAMES 構造体のハンドル
DriverName	デバイスドライバ名を取得する変数
DeviceName	プリンタ名を取得する変数
OutputName	出力ポート名を取得する変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
成功した場合、プリンタ名などの文字列情報が DeviceName などに設定されます。
DEVNAMES 構造体については WindowsAPI 関連の書籍などをご覧ください。

IKGetImageFromHdc

【機能】
スクリーンのデバイスコンテキストに描画されたイメージを取得します。

【関数書式】
(1)C++Builder
HANDLE IKGetImageFromHdc(HDC hDC, long Width, long Height, short BitCount);
(2)Delphi
function IKGetImageFromHdc(hDC: HDC; Width, Height: Longint; BitCount: Smallint): THandle;

【引数】

名称	内容
hDC	スクリーンのデバイスコンテキスト
Width	取得するイメージの幅(ピクセル)
Height	取得するイメージの高さ(ピクセル)
BitCount	作成したいイメージのビット数

【戻り値】
ラスタイメージのメモリハンドル(実行に失敗した場合はヌル(0)が返されます。)

【解説】
スクリーンのデバイスコンテキストに描画されたイメージを取得し、ラスタイメージのメモリハンドルとして返します。実際のイメージの幅と高さが取得できないコンポーネントからのイメージの取得は、上手くいかない場合があります。

IKGetPaperSize

【機能】
プリンタのデバイスコンテキストから印刷有効領域と用紙サイズを取得します。

【関数書式】
(1)C++Builder
BOOL IKGetPaperSize(HDC hDC, LPRECT PrintRect, LPLONG Width, LPLONG Height, BYTE UnitMode);
(2)Delphi
function IKGetPaperSize(hDC: HDC; var PrintRect: TRect; var Width, Height: Longint; UnitMode: Byte): LongBool;

【引数】

名称	内容
hDC	プリンタのデバイスコンテキスト
PrintRect	印刷有効領域を取得する構造体変数
Width	用紙の幅を取得する変数
Height	用紙の高さを取得する変数
UnitMode	取得する単位(0:ピクセル単位、1:0.1mm 単位)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
プリンタのデバイスコンテキストから印刷有効領域と用紙サイズを取得します。

PrintRect のメンバー変数と Width,Height をピクセル単位として返す場合
UnitMode が 0
PrintRect のメンバー変数と Width,Height を 0.1mm 単位として返す場合
UnitMode が 1

コード例(Delphi)

```
Ret: LongBool
Rect1: TRect;
Rect2: TRect;
Width: Longint;
Height: Longint
//用紙サイズおよび印字有効領域を取得
Ret := IKGetPaperSize(DC, Rect1, Width, Height, 1);
//有効領域いっぱいに印字
Rect2.Left := 0;
Rect2.Top := 0;
Rect2.Right := Rect1.Right - Rect1.Left;
Rect2.Bottom := Rect1.Bottom - Rect1.Top;
Ret := IKImageOut(DC, ImgHandle, Rect2, False, True, 1);
```

【ImageKit5 との違い】

関数名	引数の並び
IK5GetPaperSize:	hDC, PrintRect, Width, Height
IKGetPaperSize:	hDC, PrintRect, Width, Height, UnitMode

IK5SetDeviceMode で設定した値を引数の UnitMode に渡すと IK5 と同じ動作となります。

IKGetPixel

【機能】
指定したピクセルから RGB 値を取得します。

【関数書式】
(1)C++Builder
BOOL IKGetPixel(LPVOID DeviceValue, int x, int y, LPBYTE Red, LPBYTE Green, LPBYTE Blue, BYTE DeviceMode);
(2)Delphi
function IKGetPixel(DeviceValue: THandle; x, y: Integer; var Red, Green, Blue: Byte; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル（DeviceMode による）
x,y	ピクセルの座標
Red	RGB の赤のパレット
Green	RGB の緑のパレット
Blue	RGB の青のパレット
DeviceMode	描画対象（0:スクリーン、2:メモリハンドル）

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
x,y で指定したピクセルから RGB 値を取得します。（対象はスクリーン、メモリハンドル）
DeviceMode が 0 もしくは 2 の場合に有効になります。x,y の値はピクセル単位で設定してください。

【ImageKit5 との違い】

関数名	**引数の並び**
IK5GetPixel:	DeviceValue, x, y, Red, Green, Blue
IKGetPixel:	DeviceValue, x, y, Red, Green, Blue, DeviceMode

IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

IKGetPrinterPort

【機能】
プリンタのポートを取得します。

【関数書式】
(1)C++Builder
DWORD IKGetPrinterPort(LPCTSTR PrinterName, LPTSTR Port);
(2)Delphi
function IKGetPrinterPort(PrinterName, Port: PChar): DWORD;

【引数】

名称	内容
PrinterName	プリンタ名
Port	ポートを取得する変数

【戻り値】
Port に設定された文字列の長さを返します(終端のヌルは除く、実行に失敗した場合は 0)。

【解説】
プリンタが複数のポートに接続されている場合、各ポート名はカンマで区切られます。
例:"LPT1:,LPT2:"

IKGetTextExtent

【機能】
テキストの情報から文字列の幅と高さを取得します。

【関数書式】
(1)C++Builder
BOOL IKGetTextExtent(HDC hDC, LPLONG Width, LPLONG Height, PTR_IKPRINT_TEXTINFO TextInfo, LPCTSTR Text, BYTE UnitMode);
(2)Delphi
function IKGetTextExtent(hDC: HDC; var Width, Height: Longint; var TextInfo: IKPRINT_TEXTINFO; Text: PChar; UnitMode: Byte): LongBool;

【引数】

名称	内容
hDC	デバイスコンテキスト
Width	文字列の幅を取得する変数
Height	文字列の高さを取得する変数
TextInfo	描画情報を指定する構造体変数
Text	計測する文字列(終端はヌル)
UnitMode	計測時の単位(0:ピクセル単位、1:0.1mm 単位)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
テキストの情報から文字列の幅と高さを取得します。(単一行のみ対応)
取得するには TextInfo の **CharSet**,**FontName**,**FontSize**,**Direction**,**CharAngle**,**RotateString** に値を設定する必要があります。
CharExtra に 1 以上の値を設定すると文字間を考慮した値になります。
RotateString が 0 の場合は文字列の方向と文字の回転に関わらず、文字列を描画する横方向が幅で縦方向が高さとなります。**RotateString** が 0 以外の場合は文字列を描画する方向が幅となり、垂直方向が高さとなります。**RotateString** が 0 の場合、**CharAngle** は 0,90,180,270 以外は無効です。

Width,Height をピクセル単位として返す場合
UnitMode が 0
Width,Height を 0.1mm 単位として返す場合
UnitMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5GetTextExtent(A):	hDC, Width, Height, PrintInfo
IKGetTextExtent:	hDC, Width, Height, TextInfo, Text, UnitMode

PrintInfo と TextInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。
IK5PRINT_DATA --> IKPRINT_TEXTINFO
CharExtra,CharSet,RotateString,Alpha1,Alpha2,HotkeyPrefix が新たに追加され、Text が構造体から削除されました。
TextForeColor,TextBackColor が TextColor1,TextColor2 にそれぞれ変更されました。
IK5PRINT_DATA_A --> IKPRINT_TEXTINFO
CharSet, RotateString,Alpha1,Alpha2,HotkeyPrefix が新たに追加され、Text が構造体から削除されました。
TextForeColor,TextBackColor が TextColor1,TextColor2 にそれぞれ変更されました。
PrintInfo.Text を引数の Text に、IK5SetDeviceMode で設定した値を引数の UnitMode に、TextInfo の CharSet を-1 に、RotateString を 0 に設定すると IK5 と同じ動作となります。

【ImageKit6 との違い】
IKPRINT_TEXTINFO 構造体に Alpha1,Alpha2,HotkeyPrefix が追加され、TextForeColor,TextBackColor の名称が変更されましたが、IKGetTextExtent では使用するメンバー変数が同じなため動作に変わりはありません。

IKImageOut

【機能】
デバイスコンテキストにイメージを描画します。

【関数書式】
(1)C++Builder
BOOL IKImageOut(HDC hDC, HANDLE Handle, LPRECT PrintRect, BOOL Aspect, BOOL DXFBlack, BYTE DeviceMode);
(2)Delphi
function IKImageOut(hDC: HDC; Handle: THandle; var PrintRect: TRect; Aspect, DXFBlack: LongBool; DeviceMode: Byte): LongBool;

【引数】

名称	内容
hDC	デバイスコンテキスト
Handle	ラスタイメージもしくはベクトルイメージのメモリハンドル
PrintRect	描画する矩形を指定する構造体変数
Aspect	縦横比の設定［True(0 以外):する、False(0):しない］
DXFBlack	描画イメージが DXF の場合、白を黒に置き換えて描画［ True(0 以外):する False(0):しない ］
DeviceMode	描画対象(0:スクリーン、1:プリンタ)

【戻り値】
実行に失敗した場合はFalse(0)、成功した場合はTrue(0 以外)が返されます。(印刷処理中にダイアログボックスを表示してキャンセルした場合は、False(0)が返されます。)

【解説】
PrintRect で指定された範囲内にイメージが収まりきらない場合は、イメージをスケーリングします。
PrintRect にイメージをそのまま描画する場合は Aspect に False(0)を、イメージの縦横比と描画領域の縦横比を考慮する場合は Aspect に True(0 以外)を設定してください。
(描画対象はスクリーン、プリンタ)

PrintRect のメンバー変数をピクセル単位として扱う場合
DeviceMode が 0
PrintRect のメンバー変数を 0.1mm 単位として扱う場合
DeviceMode が 1

コード例(Delphi)

```
Ret: LongBool
Rect1: TRect;
Rect2: TRect;
Width: Longint;
Height: Longint
//用紙サイズおよび印字有効領域を取得
Ret := IKGetPaperSize(DC, Rect1, Width, Height, 1);
//有効領域いっぱいに印字
Rect2.Left := 0;
Rect2.Top := 0;
Rect2.Right := Rect1.Right - Rect1.Left;
Rect2.Bottom := Rect1.Bottom - Rect1.Top;
Ret := IKImageOut(DC, ImgHandle, Rect2, False, True, 1);
```

【ImageKit5 との違い】

関数名	引数の並び
IK5ImageOut:	hDC, Handle, PrintRect, Aspect
IKImageOut:	hDC, Handle, PrintRect, Aspect, DXFBlack, DeviceMode

DXFBlack に False(0)を与え、IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

IKImageOutToHwnd

【機能】
ウィンドウハンドルに対してイメージを描画します。

【関数書式】
(1)C++Builder
BOOL IKImageOutToHwnd(HWND hWnd, HANDLE Handle, COLORREF Color, BOOL Aspect, BOOL DXFBlack);
(2)Delphi
function IKImageOutToHwnd(hWnd: HWND; Handle: THandle; Color: COLORREF; Aspect, DXFBlack: LongBool): LongBool;

【引数】

名称	内容
hWnd	ウィンドウハンドル
Handle	ラスタイメージもしくはベクトルイメージのメモリハンドル
Color	ウィンドウの背景色
Aspect	縦横比の設定［True(0 以外):する、False(0):しない］
DXFBlack	描画イメージが DXF の場合、白を黒に置き換えて描画［ True(0 以外):する　False(0):しない ］

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
イメージがウィンドウの描画領域に収まりきらない場合は、イメージをスケーリングします。
ウィンドウの描画領域にイメージをそのまま描画する場合は Aspect に False(0)を、イメージの縦横比と描画領域の縦横比を考慮する場合は Aspect に True(0 以外)を設定してください。
(描画対象はスクリーン)
Color には **RGB**(Red,Green,Blue)として求めた値などを設定してください。

【ImageKit5 との違い】

関数名	**引数の並び**
IK5ImageOutToHwnd:	hDC, Handle, Color, Aspect
IKImageOutToHwnd:	hDC, Handle, Color, Aspect, DXFBlack

IKLine

【機能】
描画先オブジェクトに直線を描画します。

【関数書式】
(1)C++Builder
BOOL IKLine(LPVOID DeviceValue, LPRECT PrintRect, PTR_IKPRINT_DRAWINFO DrawInfo、BYTE DeviceMode);
(2)Delphi
function IKLine(DeviceValue: THandle; var PrintRect: TRect; var DrawInfo: IKPRINT_DRAWINFO; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
PrintRect	始点(left,top)、終点(right,bottom)を指定する構造体変数
DrawInfo	描画する情報を指定する構造体変数
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PrintRect の left,top で指定された座標から、right,bottom で指定された座標まで直線を描画します。ただし、right,bottom の点自体は線に含まれず描画されません。(対象はスクリーン、プリンタ、メモリハンドル)
描画するには DrawInfo の **PenWidth**,**PenStyle**,**PenMode**,**PenColor**,**Transparent**,**BackColor** に値を設定する必要があります。
BackColor は **Transparent** = 0 でペンが実線以外の場合に有効です。

PrintRect のメンバー変数をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
PrintRect のメンバー変数を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5Line:	DeviceValue, PrintRect, PrintInfo
IKLine:	DeviceValue, PrintRect, DrawInfo, DeviceMode

PrintInfo と DrawInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。Transparent を 0 以外に設定し、IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

【ImageKit6 との違い】
IKPRINT_DRAWINFO 構造体に BackColor が追加されました。Transparent が 0 以外であれば IK6 と同じ動作です。

IKMeasureString

【機能】
GDI+の機能を利用してテキストの情報から文字列の幅と高さを取得します。

【関数書式】
(1)C++Builder
BOOL IKMeasureString(HDC hDC, int AreaWidth, int AreaHeight, LPFLOAT Width, LPFLOAT Height, LPINT charactersFitted, LPINT linesFilled, int FormatFlags, PTR_IKPRINT_TEXTINFO TextInfo, LPCTSTR Text, BYTE UnitMode);
(2)Delphi
function IKMeasureString(hDC: HDC; AreaWidth, AreaHeight: Integer; var Width, Height: Single; var charactersFitted, linesFilled: Integer; FormatFlags: Integer; var TextInfo: IKPRINT_TEXTINFO; Text: PChar; UnitMode: Byte): LongBool;

【引数】

名称	内容
hDC	デバイスコンテキスト
AreaWidth	矩形領域の最大幅(0～)
AreaHeight	矩形領域の最大高さ(0～)
Width	取得する文字列の幅
Height	取得する文字列の高さ
charactersFitted	取得する文字列の文字数
linesFilled	取得する文字列の行数
FormatFlags	文字列の書式情報(デフォルトは 0)
TextInfo	描画情報を指定する構造体
Text	計測する文字列(終端はヌル)
UnitMode	計測時の単位(0:ピクセル単位、1:0.1mm 単位)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
文字列の幅と高さを取得するには TextInfo の **FontName**,**FontSize**,**HCentering**,**VCentering**,**HotkeyPrefix** に値を設定する必要があります。AreaWidth と AreaHeight のどちらかに 0 を設定した場合は、Text で与えられた文字列の幅と高さを取得しますが、AreaWidth と AreaHeight に 1 以上を与えた場合は、その矩形領域に含まれる文字列の幅と高さを取得します。charactersFitted と linesFilled についても同様な扱いとなります。
当関数を実行する場合は、事前に IKPrintGdipStart 関数を実行する必要があります。

AreaWidth,AreaHeight,Width,Height をピクセル単位として扱う場合
UnitMode が 0
AreaWidth,AreaHeight,Width,Height を 0.1mm 単位として扱う場合
UnitMode が 1

文字列の書式情報を示します。FormatFlags に設定する値は次の値の論理和となります。

値	説明
0x00000001	テキストの方向を右から左に指定します。
0x00000002	テキストが縦書きになるよう指定します。
0x00000004	グリフが外接する四角形にかからないよう指定します。既定では、端に表示する必要がある場合、一部のグリフが若干四角形にかかるよう設定されます。
0x00000020	左から右を指示するマークなどの制御文字をグリフで表現します。
0x00000400	フォールバックを無効にして、要求されたフォントでサポートされていない文字のフォントを切り替えます。欠落文字は、グリフの欠落したフォント(通常は空白の正方形)で表示されます。
0x00000800	既定では、各行末の空白が除外されます。各行末の空白を計測に含める場合はこの値を設定します。

0x00001000	四角形内の書式指定時に、行間のテキストのラップを無効にします。この値は、四角形ではなく点が渡された場合、または長さゼロの行の四角形が指定された場合に暗黙的に指定されます。
0x00002000	書式指定用の四角形には、完全な直線だけがレイアウトされます。既定では、クリッピングの結果、テキストの末尾が表示された状態、または行が表示されなくなった状態のうち、いずれか早い方の状態になるまでレイアウトが継続します。既定の設定では、行高さの整数倍でない書式指定用四角形を用いた場合は、なるまでレイアウトが継続します。既定の設定では、行高さの整数倍でない書式指定用四角形を用いた場合は、最後の行の一部が隠れることがあります。必ず行全体が表示されるようにするには、この値を指定した上で、少なくとも 1 つの行と高さが同じの書式指定用の四角形をご使用ください。
0x00004000	論理グリフの突出部と書式指定用の四角形からはみ出すラップされていないテキストを表示できます。既定では、書式指定用の四角形からはみ出たテキストとグリフ部はすべてクリップされます。

※16 進表記のため、Delphi は 0x を$に置き換えてください。

コード例：

(1)C++Builder

```
IKPRINT_TEXTINFO TextInfo;
float Str_Width, Str_Height;
int charactersFitted, linesFilled;
RECT PrintRect;
ULONG_PTR token;//DWORD token;

token = IKPrintGdipStart();
if (token == 0) return;

memset(&TextInfo, 0, sizeof(IKPRINT_TEXTINFO));
TextInfo.FontSize = 20;
lstrcpy(TextInfo.FontName, "MS ゴシック");
TextInfo.HotkeyPrefix = 0;
IKMeasureString(hDC, 0, 0, &Str_Width, &Str_Height, &charactersFitted, &linesFilled, 0, &TextInfo, "テキスト", 0);

PrintRect.left = 10;
PrintRect.top = 10;
PrintRect.right = PrintRect.left + (long)Str_Width - 1;
PrintRect.bottom = PrintRect.top + (long)Str_Height - 1;
TextInfo.CharAngle = 0;
TextInfo.TextColor1 = RGB(255, 0, 0);
TextInfo.Alpha1 = 255;
TextInfo.TextColor2 = RGB(255, 255, 255);
TextInfo.Alpha2 = 255;
//グラデーションブラシを使用
IKDrawString(hDC, &PrintRect, 4, 0, 0, 0, 0, &TextInfo, "テキスト", 0);
IKPrintGdipEnd(token)
```

(2)Delphi

```
TextInfo: IKPRINT_TEXTINFO;
Str_Width, Str_Height: Single;
CharactersFitted, linesFilled: Integer;
PrintRect: TRect;
token: DWORD;

token := IKPrintGdipStart();
if token = 0 then Exit;

FillChar(TextInfo, SizeOf(TextInfo), 0);
TextInfo.FontSize := 20;
StrPCopy(TextInfo.FontName, 'MS ゴシック');
TextInfo.HotkeyPrefix := 0;
IKMeasureString(DC, 0, 0, Str_Width, Str_Height, charactersFitted, linesFilled, 0, TextInfo, 'テキスト', 0);

PrintRect.Left := 10;
```

```
PrintRect.Top := 10;
PrintRect.Right := PrintRect.Left + Trunc(Str_Width) - 1;
PrintRect.Bottom := PrintRect.Top + Trunc(Str_Height) - 1;
TextInfo.CharAngle := 0;
TextInfo.TextColor1 := clRed;
TextInfo.Alpha1 := 255;
TextInfo.TextColor2 := clWhite;
TextInfo.Alpha2 := 255;
//グラデーションブラシを使用
IKDrawString(DC, PrintRect, 4, 0, 0, 0, 0, TextInfo, 'テキスト', 0);
IKPrintGdipEnd(token)
```

IKPaint

【機能】
描画先オブジェクトの指定した点を基準に別の色で塗りつぶします。

【関数書式】
(1)C++Builder
BOOL IKPaint(LPVOID DeviceValue, int x, int y, COLORREF FColor, COLORREF TColor, BYTE PaintMode, BYTE DeviceMode);
(2)Delphi
function IKPaint(DeviceValue: THandle; x, y: Integer; FColor, TColor: COLORREF; PaintMode, DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
x,y	開始座標
FColor	塗りつぶされた領域色、もしくは囲まれた領域の境界色
TColor	塗りつぶす色
PaintMode	描画方法
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PaintMode が 0 の場合
x,y を開始点として FColor を境界色として囲まれた領域の内部(外部)を TColor で塗りつぶします。
PaintMode が 1 の場合
x,y を開始点として FColor で塗りつぶされた領域を TColor で塗りつぶします。

x,y から FColor に突き当たるまで全ての方向に塗りつぶしが行われます。
(対象はスクリーン、プリンタ、メモリハンドル)
FColor,TColor には **RGB**(Red,Green,Blue)として求めた値などを設定してください。

x,y をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
x,y を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5Paint:	DeviceValue, x, y, FColor, TColor
IKPaint:	DeviceValue, x, y, FColor, TColor, PaintMode, DeviceMode

IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

IKPie

【機能】
描画先オブジェクトに扇形を描画します。

【関数書式】
(1)C++Builder
BOOL IKPie(LPVOID DeviceValue, LPRECT PrintRect, int x1, int y1, int x2, int y2, PTR_IKPRINT_DRAWINFO DrawInfo, BYTE DeviceMode);
(2)Delphi
function IKPie(DeviceValue: THandle; var PrintRect: TRect; x1, y1, x2, y2: Integer; var DrawInfo: IKPRINT_DRAWINFO; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
PrintRect	楕円を定義する境界矩形を指定する構造体変数
x1,y1	楕円弧の始点の x,y 座標
x2,y2	楕円弧の終点の x,y 座標
DrawInfo	描画する情報を指定する構造体変数
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
楕円の一部分を描画し、2 つの端点と楕円の中心を接続して扇形を作成します。アウトラインは **PenStyle** の値で描画され、内部は **BrushStyle** の値で塗りつぶされます。端点が円弧の上にある必要はなく、中心から指定の端点まで引いた直線との交点が計算され、その交点が用いられます。(描画対象はスクリーン、プリンタ、メモリハンドル)
描画するには DrawInfo の **PenWidth**,**PenStyle**,**PenMode**,**PenColor**,**BrushStyle**,**BrushColor**,**Transparent**,**BackColor** に値を設定する必要があります。**BackColor** は **Transparent** = 0 でペンが実線以外、あるいはブラシがハッチパターンの場合に有効です。

PrintRect のメンバー変数と x1,y1,x2,y2 の値をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
PrintRect のメンバー変数と x1,y1,x2,y2 の値を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5Pie:	DeviceValue, PrintRect, x1, y1, x2, y2, PrintInfo
IKPie:	DeviceValue, PrintRect, x1, y1, x2, y2, DrawInfo, DeviceMode

PrintInfo と DrawInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。Transparent を 0 以外に設定し、IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

【ImageKit6 との違い】
IKPRINT_DRAWINFO 構造体に BackColor が追加されました。Transparent が 0 以外であれば IK6 と同じ動作です。

IKPolyBezier

【機能】
描画先オブジェクトにベジェ曲線を描画します。

【関数書式】
(1)C++Builder
BOOL IKPolyBezier(LPVOID DeviceValue, LPPOINT lpPoint, DWORD Points, PTR_IKPRINT_DRAWINFO DrawInfo, BYTE DeviceMode);
(2)Delphi
function IKPolyBezier(DeviceValue: THandle; var lpPoint: TPoint; Points: DWORD; var DrawInfo: IKPRINT_DRAWINFO; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
lpPoint	ベジェ曲線の端点と制御点の座標を指定する構造体変数の配列 (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数
DrawInfo	描画する情報を指定する構造体変数
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
ベジェ曲線は始点、2 つの制御点、終点の 4 つの点から構成されます。最初の曲線は 4 つの点で描画されますが、それ以降の曲線は 3 つの点で描画されます。(描画対象はスクリーン、プリンタ、メモリハンドル)
描画するには DrawInfo の **PenWidth,PenStyle,PenMode,PenColor,Transparent,BackColor** に値を設定する必要があります。
BackColor は **Transparent** = 0 でペンが実線以外の場合に有効です。
Points は「3×曲線の数＋最初の始点」の数だけ必要です。

lpPoint の配列のメンバー変数の値をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
lpPoint の配列のメンバー変数の値を 0.1mm 単位として扱う場合
DeviceMode が 1

コード例:
(1)C++Builder

```
IKPRINT_DRAWINFO DrawInfo;
POINT PtCurve[4];

PtCurve[0].x = 10; PtCurve[0].y = 100;
PtCurve[1].x = 50; PtCurve[1].y = 50;
PtCurve[2].x = 100; PtCurve[2].y = 150;
PtCurve[3].x = 150; PtCurve[3].y = 100;
memset(&DrawInfo, 0, sizeof(IKPRINT_DRAWINFO));
DrawInfo.PenColor = RGB(0, 0, 255);
DrawInfo.PenStyle = 1;
DrawInfo.PenWidth = 15;
IKPolyBezier(hDC, PtCurve, 4, DrawInfo, 0);
```

(2)Delphi

```
DrawInfo: IKPRINT_DRAWINFO;
```

```
PtCurve: array[0..3] of TPoint;

PtCurve[0].x := 10; PtCurve[0].y := 100;
PtCurve[1].x := 50; PtCurve[1].y := 50;
PtCurve[2].x := 100; PtCurve[2].y := 150;
PtCurve[3].x := 150; PtCurve[3].y := 100;
FillChar(DrawInfo, SizeOf(DrawInfo), 0);
DrawInfo.PenColor := clGreen;
DrawInfo.PenStyle := 1;
DrawInfo.PenWidth := 15;
IKPolyBezier(hDC, PtCurve[0], 4, DrawInfo, 0);
```

IKPolygon

【機能】
描画先オブジェクトに多角形を描画します。

【関数書式】
(1)C++Builder
BOOL IKPolygon(LPVOID DeviceValue, LPPOINT lpPoint, int Points, PTR_IKPRINT_DRAWINFO DrawInfo, BYTE DeviceMode);
(2)Delphi
function IKPolygon(DeviceValue: THandle; var lpPoint: TPoint; Points: Integer; var DrawInfo: IKPRINT_DRAWINFO; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
lpPoint	多角形の各頂点の座標を指定する構造体変数の配列 (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(2 以上)
DrawInfo	描画する情報を指定する構造体変数
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
指定の点を基にして一連の直線を描画し、最後の点から最初の点まで直線を引くことで図形をクローズします。アウトラインは **PenStyle** の値で描画され、内部は **BrushStyle** の値で塗りつぶされます。
(描画対象はスクリーン、プリンタ、メモリハンドル)
描画するには DrawInfo の **PenWidth**,**PenStyle**,**PenMode**,**PenColor**,**BrushStyle**,**BrushColor**,**Transparent**,**BackColor** に値を設定する必要があります。**BackColor** は **Transparent** = 0 でペンが実線以外、あるいはブラシがハッチパターンの場合に有効です。

lpPoint の配列のメンバー変数の値をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
lpPoint の配列のメンバー変数の値を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5Polygon:	DeviceValue, lpPoint, Points, PrintInfo
IKPolygon:	DeviceValue, lpPoint, Points, DrawInfo, DeviceMode

PrintInfo と DrawInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。Transparent を 0 以外に設定し、IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

【ImageKit6 との違い】
IKPRINT_DRAWINFO 構造体に BackColor が追加されました。Transparent が 0 以外であれば IK6 と同じ動作です。

IKPolyline

【機能】
描画先オブジェクトに連続線を描画します。

【関数書式】
(1)C++Builder
BOOL IKPolyline(LPVOID DeviceValue, LPPOINT lpPoint, int Points, PTR_IKPRINT_DRAWINFO DrawInfo, BYTE DeviceMode);
(2)Delphi
function IKPolyline(DeviceValue: THandle; var lpPoint: TPoint; Points: Integer; var DrawInfo: IKPRINT_DRAWINFO; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
lpPoint	連続線の座標を指定する構造体変数の配列 (1)C++Builder　構造体の配列の先頭のポインタを渡す (2)Delphi　Pt: array [0..2] of TPoint の場合は、引数に Pt[0]を与える
Points	lpPoint の配列の数(2 以上)
DrawInfo	描画する情報を指定する構造体変数
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
指定の点で一連の直線を **PenStyle** の値で描画します。(描画対象はスクリーン、プリンタ、メモリハンドル)
描画するには DrawInfo の **PenWidth,PenStyle,PenMode,PenColor,Transparent,BackColor** に値を設定する必要があります。
BackColor は **Transparent** = 0 でペンが実線以外の場合に有効です。

lpPoint の配列のメンバー変数の値をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
lpPoint の配列のメンバー変数の値を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5Polyline:	DeviceValue, lpPoint, Points, PrintInfo
IKPolyline:	DeviceValue, lpPoint, Points, DrawInfo, DeviceMode

PrintInfo と DrawInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。Transparent を 0 以外に設定し、IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

【ImageKit6 との違い】
IKPRINT_DRAWINFO 構造体に BackColor が追加されました。Transparent が 0 以外であれば IK6 と同じ動作です。

IKPreviewInit

【機能】
印刷プレビュー時に、スクリーンの解像度を出力するプリンタの解像度にスケーリングします。

【関数書式】
(1)C++Builder
BOOL IKPreviewInit(HDC phDC, HDC dhDC);
(2)Delphi
function IKPreviewInit(phDC, dhDC: HDC): LongBool;

【引数】

名称	内容
phDC	出力するプリンタのデバイスコンテキスト
dhDC	プレビューするスクリーンのデバイスコンテキスト

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
印刷プレビュー時に、スクリーンの解像度を出力するプリンタの解像度にスケーリングします。

IKPrintAbortDoc

【機能】
現在の印刷ジョブを終了します。

【関数書式】
(1)C++Builder
BOOL IKPrintAbortDoc(HDC hDC);
(2)Delphi
function IKPrintAbortDoc(hDC: HDC): LongBool;

【引数】

名称	内容
hDC	**IKPrintCreateDC(Ex)**もしくは **IKPrintDialog,IKPrintDlg** で取得したデバイスコンテキスト

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
現在の印刷ジョブを終了し、最後に **IKPrintStartDoc** 関数を呼び出した後に描画したすべての情報を消去します。
IKPrintAbortDoc 関数は、**IKPrintStartDoc**,**IKPrintStartPage**,**IKPrintEndPage**,**IKPrintEndDoc** 関数と共に使用し、**IKPrintStartPage** と **IKPrintEndPage** 関数の間に記述します。
ただし、**IKPrintStartDoc** 関数の引数 ButtonName,Caption,Message などを設定して印刷中止のダイアログボックスを表示する場合は、実行する必要はありません。

印刷ジョブを終了する例:
(1)C++Builder

```
HDC hDC;
int i;
BOOL bAbort = TRUE;

hDC = IKPrintCreateDC(NULL, "Default", NULL, 1);
if (hDC == NULL) return;

if (IKPrintStartDoc(hDC, Form1->Handle, "", "", "", "ImageKit Print Sample") != FALSE) {
   for (i = 0; i < 100; i++) {
      IKPrintStartPage(hDC);

      //イメージ・テキストなどの描画

      if (i == 50) {
         IKPrintAbortDoc(hDC);
         bAbort = TRUE;
      }
      IKPrintEndPage(hDC);
      if (bAbort) break;
   }
   IKPrintEndDoc(hDC);
}
IKPrintDeleteDC(hDC);
```

(2)Delphi

```
DC: HDC;
i: Integer;
bAbort: Boolean;
```

```
DC := IKPrintCreateDC(nil, 'Default', 0, 1);
if DC = 0 then Exit;
if (IKPrintStartDoc(DC, Form1.Handle, '', '', '', 'ImageKit Print Sample') <> False) then
begin
  bAbort := False;
  for i := 0 to 99 do
  begin
    IKPrintStartPage(DC);

    //イメージ・テキストなどの描画

    if i = 50 then
    begin
      IKPrintAbortDoc(DC);
      bAbort := True;
    end;
    IKPrintEndPage(DC);
    if bAbort then Break;
  end;
  IKPrintEndDoc(DC);
end;
IKPrintDeleteDC(DC);
```

IKPrintCreateDC

【機能】
デバイスコンテキストを作成します。

【関数書式】
(1)C++Builder
HDC IKPrintCreateDC(LPCTSTR PrinterName, LPCTSTR PrintFileName, HANDLE hDevMode, short Mode);
(2)Delphi
function IKPrintCreateDC(PrinterName, PrintFileName: PChar; hDevMode: THandle; Mode: Smallint): HDC;

【引数】

名称	内容
PrinterName	プリンタ名
PrintFileName	プリンタ設定ファイル名（**IKSaveDevModeHandle(Ex)**もしくは **IKSetPrint** で保存したファイル名）
hDevMode	DEVMODE 構造体のハンドル（**IKGetDevModeHandle(Ex)**で取得したハンドル）
Mode	デバイスコンテキストを作成する方法（0～2）

【戻り値】
デバイスコンテキスト（失敗した場合は、ヌル(0)が返されます）

【解説】
Mode が 0 の場合
PrinterName に設定されたプリンタ名からデバイスコンテキストを作成します。
PrintFileName と DevMode は無効となります。
Mode が 1 の場合
PrintFileName に設定されたファイルからデバイスコンテキストを作成します。
PrinterName と DevMode は無効となります。
Mode が 2 の場合
hDevMode に設定された DEVMODE 構造体のハンドルからデバイスコンテキストを作成します。
PrinterName と PrintFileName は無効となります。

通常使うプリンタを使用する場合は、Mode を 1 にして PrintFileName を"Default"（アルファベットの大小文字関係なし）に設定してください。
戻り値のデバイスコンテキストは、IKPrintStartDoc,IKPrintStartPage,IKPrintEndPage,IKPrintEndDoc 関数と共に使用します。なお、取得したデバイスコンテキストは IKPrintDeleteDC 関数で削除します。

拡張機能として、PrintFileName にプリンタの設定情報を保存したファイル名にプラスして印刷部数、印刷の向き、用紙サイズを順番に設定すると、保存した項目の該当する部分を変更して印刷することができます。（後ろの項目は省略可、順番を入れ替えることは不可）
設定する場合はそれぞれの項目をセミコロン(;)で区切ります。
設定ファイルに保存された情報を有効にしたい場合は、それぞれの該当する項目に 0 を設定します。

印刷の向き:()内の説明は WindowsAPI で使用する定数と同じ意味です。
1:縦(DMORIENT_PORTRAIT)　2:横(DMORIENT_LANDSCAPE)
用紙サイズ:()内の説明は WindowsAPI で使用する定数と同じ意味です。
8:A3(DMPAPER_A3)　9:A4(DMPAPER_A4)
12:B4(DMPAPER_B4)　13:B5(DMPAPER_B5)
上記以外の用紙サイズについては、ご利用のコンテナの WindowsAPI に関連する部分を参考に設定してください。

設定ファイルを"IkPrint.Ik"とした場合(Delphi)
1)通常の場合
PrintFileName := 'C:¥Test¥IkPrint.Ik';
2)拡張機能を使用した場合
A)印刷部数を 3、印刷の向きを横、用紙サイズを B5 にする
PrintFileName := 'C:¥Test¥IkPrint.Ik;3;2;13';

B)印刷部数と用紙サイズを設定ファイルから読み出し、印刷の向きを横にする
PrintFileName := 'C:¥Test¥IkPrint.Ik;0;2;0'; **or** PrintFileName := 'C:¥Test¥IkPrint.Ik;0;2';

プリンタ設定ファイル(IkPrn.Ik)を使用して印刷するコード例:
(1)C++Builder

```
HDC hDC;

hDC = IKPrintCreateDC(NULL, "C:¥¥Test¥¥IkPrn.Ik", NULL, 1);
if (hDC == NULL) return;

if (IKPrintStartDoc(hDC, Form1->Handle, "", "", "", "ImageKit Print Sample") != FALSE) {
  IKPrintStartPage(hDC);

  //イメージ・テキストなどの描画

  IKPrintEndPage(hDC);
  IKPrintEndDoc(hDC);
}
IKPrintDeleteDC(hDC);
```

(2)Delphi

```
DC: HDC;

DC := IKPrintCreateDC(nil, 'C:¥Test¥IkPrn.Ik', 0, 1);
if DC = 0 then Exit;
if (IKPrintStartDoc(DC, Form1.Handle, '', '', '', 'ImageKit Print Sample') <> False) then
begin
  IKPrintStartPage(DC);

  //イメージ・テキストなどの描画

  IKPrintEndPage(DC);
  IKPrintEndDoc(DC);
end;
IKPrintDeleteDC(DC);
```

IKPrintCreateDCEx

【機能】
デバイスコンテキストを作成します。

【関数書式】
(1)C++Builder
HDC IKPrintCreateDCEx(LPCTSTR PrinterName, LPCTSTR PrintFileName, HANDLE hDevMode, HANDLE hDevNames, short Mode);
(2)Delphi
function IKPrintCreateDCEx(PrinterName, PrintFileName: PChar; hDevMode, hDevNames: THandle; Mode: Smallint): HDC;

【引数】

名称	内容
PrinterName	プリンタ名
PrintFileName	プリンタ設定ファイル名（**IKSaveDevModeHandle(Ex)**もしくは **IKSetPrint** で保存したファイル名）
hDevMode	DEVMODE 構造体のハンドル（**IKGetDevModeHandle(Ex)**で取得したハンドル）
hDevNames	DEVNAMES 構造体のハンドル（**IKGetDevModeHandleEx** で取得したハンドル）
Mode	デバイスコンテキストを作成する方法（0～2）

【戻り値】
デバイスコンテキスト（失敗した場合は、ヌル(0)が返されます）

【解説】
Mode が 0 の場合
PrinterName に設定されたプリンタ名からデバイスコンテキストを作成します。
PrintFileName と hDevMode,hDevNames は無効となります。
Mode が 1 の場合
PrintFileName に設定されたファイルからデバイスコンテキストを作成します。
PrinterName と hDevMode,hDevNames は無効となります。
Mode が 2 の場合
hDevMode と hDevNames に設定された DEVMODE 構造体と DEVNAMES 構造体のハンドルからデバイスコンテキストを作成します。hDevNames が NULL や 0 の場合は **IKPrintCreateDC** と同じ動作になります。
PrinterName と PrintFileName は無効となります。

通常使うプリンタを使用する場合は、Mode を 1 にして PrintFileName を"Default"（アルファベットの大小文字関係なし）に設定してください。
戻り値のデバイスコンテキストは、IKPrintStartDoc,IKPrintStartPage,IKPrintEndPage,IKPrintEndDoc 関数と共に使用します。なお、取得したデバイスコンテキストは IKPrintDeleteDC 関数で削除します。

拡張機能として、PrintFileName にプリンタの設定情報を保存したファイル名にプラスして印刷部数、印刷の向き、用紙サイズを順番に設定すると、保存した項目の該当する部分を変更して印刷することができます。（後ろの項目は省略可、順番を入れ替えることは不可）
設定する場合はそれぞれの項目をセミコロン(;)で区切ります。
設定ファイルに保存された情報を有効にしたい場合は、それぞれの該当する項目に 0 を設定します。

印刷の向き:()内の説明は WindowsAPI で使用する定数と同じ意味です。
1:縦(DMORIENT_PORTRAIT)　2:横(DMORIENT_LANDSCAPE)
用紙サイズ:()内の説明は WindowsAPI で使用する定数と同じ意味です。
8:A3(DMPAPER_A3)　9:A4(DMPAPER_A4)
12:B4(DMPAPER_B4)　13:B5(DMPAPER_B5)
上記以外の用紙サイズについては、ご利用のコンテナの WindowsAPI に関連する部分を参考に設定してください。

設定ファイルを"IkPrint.Ik"とした場合(Delphi)
1)通常の場合

PrintFileName := 'C:¥Test¥IkPrint.Ik';

2)拡張機能を使用した場合

A)印刷部数を 3、印刷の向きを横、用紙サイズを B5 にする

PrintFileName := 'C:¥Test¥IkPrint.Ik;3;2;13';

B)印刷部数と用紙サイズを設定ファイルから読み出し、印刷の向きを横にする

PrintFileName := 'C:¥Test¥IkPrint.Ik;0;2;0'; or PrintFileName := 'C:¥Test¥IkPrint.Ik;0;2';

プリンタ設定ファイル(IkPrn.Ik)を使用して印刷するコード例:

(1)C++Builder

```
HDC hDC;

hDC = IKPrintCreateDCEx(NULL, "C:¥¥Test¥¥IkPrn.Ik", NULL, NULL, 1);
if (hDC == NULL) return;

if (IKPrintStartDoc(hDC, Form1->Handle, "", "", "", "ImageKit Print Sample") != FALSE) {
  IKPrintStartPage(hDC);

  //イメージ・テキストなどの描画

  IKPrintEndPage(hDC);
  IKPrintEndDoc(hDC);
}
IKPrintDeleteDC(hDC);
```

(2)Delphi

```
DC: HDC;

DC := IKPrintCreateDCEx(nil, 'C:¥Test¥IkPrn.Ik', 0, 0, 1);
if DC = 0 then Exit;
if (IKPrintStartDoc(DC, Form1.Handle, '', '', '', 'ImageKit Print Sample') <> False) then
begin
  IKPrintStartPage(DC);

  //イメージ・テキストなどの描画

  IKPrintEndPage(DC);
  IKPrintEndDoc(DC);
end;
IKPrintDeleteDC(DC);
```

IKPrintDeleteDC

【機能】
デバイスコンテキストを削除します。

【関数書式】
(1)C++Builder
BOOL IKPrintDeleteDC(HDC hDC);
(2)Delphi
function IKPrintDeleteDC(hDC: HDC): LongBool;

【引数】

名称	内容
hDC	**IKPrintCreateDC(Ex)**もしくは **IKPrintDialog,IKPrintDlg** で取得したデバイスコンテキスト

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
印刷処理が終了し、デバイスコンテキストが不要になった段階で実行します。
印刷コードについては IKPrintCreateDC(Ex)と IKPrintDialog,IKPrintDlg を参照してください。

IKPrintDialog

【機能】
デバイスコンテキストを作成します。

【関数書式】
(1)C++Builder
HDC IKPrintDialog(HWND hWnd, LPHANDLE hDevMode, LPHANDLE hDevNames, PTR_IKPRINT_DIALOG IkPrnDlg);
(2)Delphi
function IKPrintDialog(hWnd: HWND; hDevMode, hDevNames: PHandle; var IkPrnDlg: IKPRINT_DIALOG): HDC;

【引数】

名称	内容
hWnd	ウィンドウハンドル
hDevMode	DEVMODE 構造体のハンドル(**IKGetDevModeHandle(Ex)**で取得したハンドル)、または取得する変数
hDevNames	DEVNAMES 構造体のハンドル(**IKGetDevModeHandleEx** で取得したハンドル)、または取得する変数
IkPrnDlg	印刷ダイアログの設定

【戻り値】
デバイスコンテキスト(失敗した場合は、ヌル(0)が返されます)

【解説】
印刷ダイアログを表示し、設定した情報でデバイスコンテキストを作成します。
hDevMode(またはhDevModeとhDevNamesの両方)を設定すると、印刷ダイアログを表示する際にプリンタ名や印刷条件を操作できます。通常使うプリンタの印刷設定をそのまま使用する場合は、hDevModeとhDevNamesにNULLやnilまたは引数として渡す変数に0を設定してください。当関数実行後にデバイスコンテキスト作成時の情報を取得する場合はhDevModeとhDevNamesに渡す引数にNULLやnilを設定しないでください。デバイスコンテキストが作成されるとhDevModeとhDevNamesが更新されます。IKGetDevModeHandleEx関数で取得したハンドルをhDevModeとhDevNamesに渡すのではなく、当関数で両方のハンドルを取得する場合は引数として渡す変数に0を設定してください。また、その場合は印刷処理終了後にhDevModeとhDevNamesをIKReleaseDevModeHandleEx関数で解放してください。
IkPrnDlgは印刷ダイアログを表示する際の初期値として使用され、処理終了後には設定した情報が返されます。
IKPRINT_DIALOGについては**「Ik10Print.dll/Ik10PrintA.dll/Ik10Print64.dll/Ik10Print64A.dll」の構造体の定義**をご覧ください。
戻り値のデバイスコンテキストは、IKPrintStartDoc関数などで使用します。なお、取得したデバイスコンテキストはIKPrintDeleteDC関数で削除します。

印刷コード例:
(1)C++Builder

```
HDC hDC;
IKPRINT_DIALOG IkPrnDlg;
HANDLE hDevMode, hDevNames;

hDevMode = hDevNames = NULL;
hDevMode = IKGetDevModeHandleEx("", "Default", &hDevNames);

memset(&IkPrnDlg, 0x00, sizeof(IkPrnDlg));
IkPrnDlg.Copies = 1;
IkPrnDlg.Options = 0x4 | 0x8 | 0x100000
hDC = IKPrintDialog(Form1->Handle, &hDevMode, &hDevNames, &IkPrnDlg);
if (hDC == NULL) return;
if (IKPrintStartDoc(hDC, Form1->Handle, "", "", "", "ImageKit Print Sample") != FALSE) {
  IKPrintStartPage(hDC);

  //イメージ・テキストなどの描画
```

```
        IKPrintEndPage(hDC);
        IKPrintEndDoc(hDC);
    }
    IKPrintDeleteDC(hDC);

    IKReleaseDevModeHandleEx(hDevMode, hDevNames);
```

(2)Delphi

```
    DC: HDC;
    IkPrnDlg: IKPRINT_DIALOG;
    hDevMode, hDevNames: THandle;

    hDevMode := nil;
    hDevNames := nil;
    hDevMode := IKGetDevModeHandleEx('', 'Default', @hDevNames);

    FillChar(IkPrnDlg, SizeOf(IkPrnDlg), 0);
    IkPrnDlg.Copies := 1;
    IkPrnDlg.Options := $4 or $8 or $100000
    DC := IKPrintDialog(Form1.Handle, @hDevMode, @hDevNames, IkPrnDlg);
    if DC = 0 then Exit;
    if (IKPrintStartDoc(DC, Form1.Handle, '', '', '', 'ImageKit Print Sample') <> False) then
    begin
        IKPrintStartPage(DC);

        //イメージ・テキストなどの描画

        IKPrintEndPage(DC);
        IKPrintEndDoc(DC);
    end;
    IKPrintDeleteDC(DC);

    IKReleaseDevModeHandleEx(hDevMode, hDevNames);
```

IKPrintDlg

【機能】
デバイスコンテキストを作成します。

【関数書式】
(1)C++Builder
HDC IKPrintDlg(HWND hWnd, HANDLE hDevMode, PTR_IKPRINT_DIALOG IkPrnDlg);
(2)Delphi
function IKPrintDlg(hWnd: HWND; hDevMode: THandle; var IkPrnDlg: IKPRINT_DIALOG): HDC;

【引数】

名称	内容
hWnd	ウィンドウハンドル
hDevMode	DEVMODE 構造体のハンドル（**IKGetDevModeHandle(Ex)**で取得したハンドル）
IkPrnDlg	印刷ダイアログの設定

【戻り値】
デバイスコンテキスト（失敗した場合は、ヌル(0)が返されます）

【解説】
印刷ダイアログを表示し、設定した情報でデバイスコンテキストを作成します。
hDevMode を設定すると、印刷ダイアログを表示する際にプリンタ名や印刷条件を操作できます。
通常使うプリンタの印刷設定をそのまま使用する場合は、hDevMode に NULL もしくは 0 を設定してください。
IkPrnDlg は印刷ダイアログを表示する際の初期値として使用され、処理終了後には設定した情報が返されます。
IKPRINT_DIALOG については**「Ik10Print.dll/Ik10PrintA.dll/Ik10Print64.dll/Ik10Print64A.dll」の構造体の定義**をご覧ください。
戻り値のデバイスコンテキストは、IKPrintStartDoc 関数などで使用します。なお、取得したデバイスコンテキストは IKPrintDeleteDC 関数で削除します。

印刷コード例:
(1)C++Builder

```
HDC hDC;
IKPRINT_DIALOG IkPrnDlg;

memset(&IkPrnDlg, 0x00, sizeof(IkPrnDlg));
IkPrnDlg.Copies = 1;
IkPrnDlg.Options = 0x4 | 0x8 | 0x100000
hDC = IKPrintDlg(Form1->Handle, NULL, &IkPrnDlg);
if (hDC == NULL) return;
if (IKPrintStartDoc(hDC, Form1->Handle, "", "", "", "ImageKit Print Sample") != FALSE) {
  IKPrintStartPage(hDC);

  //イメージ・テキストなどの描画

  IKPrintEndPage(hDC);
  IKPrintEndDoc(hDC);
}
IKPrintDeleteDC(hDC);
```

(2)Delphi

```
DC: HDC;
IkPrnDlg: IKPRINT_DIALOG;

FillChar(IkPrnDlg, SizeOf(IkPrnDlg), 0);
IkPrnDlg.Copies := 1;
```

```
IkPrnDlg.Options := $4 or $8 or $100000
DC := IKPrintDlg(Form1.Handle, 0, IkPrnDlg);
if DC = 0 then Exit;
if (IKPrintStartDoc(DC, Form1.Handle, '', '', '', 'ImageKit Print Sample') <> False) then
begin
  IKPrintStartPage(DC);

  //イメージ・テキストなどの描画

  IKPrintEndPage(DC);
  IKPrintEndDoc(DC);
end;
IKPrintDeleteDC(DC);
```

IKPrintEndDoc

【機能】
印刷を終了します。

【関数書式】
(1)C++Builder
BOOL IKPrintEndDoc(HDC hDC);
(2)Delphi
function IKPrintEndDoc(hDC: HDC): LongBool;

【引数】

名称	内容
hDC	**IKPrintDialog,IKPrintDlg** もしくは **IKPrintCreateDC(Ex)**で取得したデバイスコンテキスト

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
IKPrintEndDoc は、IKPrintStartDoc,IKPrintStartPage,IKPrintEndPage と共に使用します。
印刷コードについては IKPrintCreateDC(Ex)と IKPrintDialog,IKPrintDlg を参照してください。

【ImageKit5 との違い】
戻り値が新たに追加されました。

IKPrintEndPage

【機能】
ページ単位の印刷を終了して、改ページを行います。

【関数書式】
(1)C++Builder
BOOL IKPrintEndPage(HDC hDC);
(2)Delphi
function IKPrintEndPage(hDC: HDC): LongBool;

【引数】

名称	内容
hDC	**IKPrintDialog,IKPrintDlg** もしくは **IKPrintCreateDC(Ex)**で取得したデバイスコンテキスト

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
IKPrintEndPage は、IKPrintStartDoc,IKPrintStartPage,IKPrintEndDoc と共に使用します。
印刷コードについては IKPrintCreateDC(Ex)と IKPrintDialog,IKPrintDlg を参照してください。

【ImageKit5 との違い】
戻り値が新たに追加されました。

IKPrintGdipEnd

【機能】
GDI+の終了処理を行います。

【関数書式】
(1)C++Builder
void IKPrintGdipEnd(ULONG_PTR token);
(2)Delphi
procedure IKPrintGdipEnd(token: DWORD);

【引数】

名称	内容
token	IKPrintGdipStart で取得したトークン

【戻り値】
ありません。

【解説】
IKDrawString や IKMeasureString 関数を使用する場合に使用します。

IKPrintGdipStart

【機能】
GDI+の初期化処理を行います。

【関数書式】
(1)C++Builder
ULONG_PTR IKPrintGdipStart(void);
(2)Delphi
function IKPrintGdipStart(): DWORD;

【引数】
ありません。

【戻り値】
トークン（失敗した場合は、0 が返されます）。

【解説】
IKDrawString や IKMeasureString 関数を使用する場合に使用します。

IKPrintGetArrayNum

【機能】
指定されたプリンタ名からサポートされている項目を取得する際に必要な配列の要素数を取得します。

【関数書式】
(1)C++Builder
```
int IKPrintGetArrayNum(LPCTSTR PrinterName, short CapNo);
```
(2)Delphi
```
function IKPrintGetArrayNum(PrinterName: PChar; CapNo: Smallint): Integer;
```

【引数】

名称	内容
PrinterName	プリンタ名
CapNo	取得する項目の種類(0:用紙サイズ、1:用紙トレイ、2:解像度)

【戻り値】
取得する配列に必要な要素数(0 は失敗)

【解説】
IKEnumPaperSizes,IKEnumPaperBins,IKEnumResolutions 関数で引数として渡す配列の要素数を取得します。

コード例:
(1)C++Builder
```
int Size;
LPPOINT Resolutions;

Size := IKPrintGetArrayNum("EPSON LP-8200C", 2);
if (Size < 1) return;
Resolutions = (LPPOINT)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, sizeof(POINT) * Size);
_try {
   IKEnumResolutions("EPSON LP-8200C", Resolutions);
   // 様々な処理
   ・・・・・・・
} _finally {
   HeapFree(GetProcessHeap(), 0, Resolutions);
}
```
(2)Delphi
```
Size: Integer;
Resolutions: array of TPoint;

Size := IKPrintGetArrayNum('EPSON LP-8200C', 2);
if Size < 1 then Exit;
SetLength(Resolutions, Size);
IKEnumResolutions('EPSON LP-8200C', Resolutions[0]);
// 様々な処理
・・・・・・・
```

IKPrintStartDoc

【機能】
印刷を開始します。

【関数書式】
(1)C++Builder
BOOL IKPrintStartDoc(HDC hDC, HWND hWnd, LPCTSTR Caption, LPCTSTR Message, LPCTSTR ButtonName, LPCTSTR DocName);
(2)Delphi
function IKPrintStartDoc(hDC: HDC; hWnd: HWND; Caption, Message, ButtonName, DocName: PChar): LongBool;

【引数】

名称	内容
hDC	**IKPrintDialog,IKPrintDlg** もしくは **IKPrintCreateDC(Ex)**で取得したデバイスコンテキスト
hWnd	ウィンドウハンドル
Caption	印刷中止ダイアログのキャプションに表示する文字列
Message	印刷中止ダイアログの中央に表示する文字列
ButtonName	印刷中止ダイアログのボタンに表示する文字列
DocName	印刷するドキュメント名を表す文字列

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
Caption,ButtonName,Message が空の文字列の場合は印刷中止のダイアログボックスは表示されません。IKPrintStartDoc は、IKPrintStartPage,IKPrintEndPage,IKPrintEndDoc と共に使用します。
印刷コードについては IKPrintCreateDC(Ex)と IKPrintDialog,IKPrintDlg を参照してください。

【ImageKit5 との違い】

関数名	**引数の並び**
IK5PrintStartDoc:	hWnd, Caption, Message, ButtonName, PrintFileName
IKPrintStartDoc:	hDC, hWnd, Caption, Message, ButtonName, DocName

印刷するドキュメント名が Message から DocName に変更されました。
戻り値についても IK5 ではデバイスコンテキストを返していましたが、IK6 以降では成功もしくは失敗を返すようになりました。
IK6 以降では IKPrintDialog,IKPrintDlg もしくは IKPrintCreateDC(Ex)を使用してデバイスコンテキストを取得します。

IKPrintStartPage

【機能】
ページ単位の印刷を開始します。

【関数書式】
(1)C++Builder
BOOL IKPrintStartPage(HDC hDC);
(2)Delphi
function IKPrintStartPage(hDC: HDC): LongBool;

【引数】

名称	内容
hDC	**IKPrintDialog,IKPrintDlg** もしくは **IKPrintCreateDC(Ex)**で取得したデバイスコンテキスト

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
IKPrintStartPage は、IKPrintStartDoc,IKPrintEndPage,IKPrintEndDoc と共に使用します。
印刷コードについては IKPrintCreateDC(Ex)と IKPrintDialog,IKPrintDlg を参照してください。

IKRectangle

【機能】
描画先オブジェクトに矩形を描画します。

【関数書式】
(1)C++Builder
BOOL IKRectangle(LPVOID DeviceValue, LPRECT PrintRect, PTR_IKPRINT_DRAWINFO DrawInfo, BYTE DeviceMode);
(2)Delphi
function IKRectangle(DeviceValue: THandle; var PrintRect: TRect; var DrawInfo: IKPRINT_DRAWINFO; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
PrintRect	矩形を指定する構造体変数
DrawInfo	描画する情報を指定する構造体変数
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PrintRect で指定した座標を基に矩形を描画します。アウトラインは **PenStyle** の値で描画され、内部は **BrushStyle** の値で塗りつぶされます。(対象はスクリーン、プリンタ、メモリハンドル)
描画するには DrawInfo の **PenWidth**,**PenStyle**,**PenMode**,**PenColor**,**BrushStyle**,**BrushColor**,**Transparent**,**BackColor** に値を設定する必要があります。**BackColor** は **Transparent** = 0 でペンが実線以外、あるいはブラシがハッチパターンの場合に有効です。

PrintRect のメンバー変数の値をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
PrintRect のメンバー変数の値を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5Rectangle:	DeviceValue, PrintRect, PrintInfo
IKRectangle:	DeviceValue, PrintRect, DrawInfo, DeviceMode

PrintInfo と DrawInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。Transparent を 0 以外に設定し、IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

【ImageKit6 との違い】
IKPRINT_DRAWINFO 構造体に BackColor が追加されました。Transparent が 0 以外であれば IK6 と同じ動作です。

IKReleaseDevModeHandle

【機能】
DEVMODE 構造体や DEVNAMES 構造体のハンドルを解放します。

【関数書式】
(1)C++Builder
BOOL IKReleaseDevModeHandle(HANDLE hDevMode);
(2)Delphi
function IKReleaseDevModeHandle(hDevMode: THandle): LongBool;

【引数】

名称	内容
hDevMode	**IKGetDevModeHandle(Ex)や IKPrintDialog** で取得した DEVMODE や DEVNAMES 構造体のハンドル

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
DEVMODE 構造体や DEVNAMES 構造体のハンドルが不要になった段階で実行します。
成功した場合、hDevMode が占有するメモリは解放されます。

IKReleaseDevModeHandleEx

【機能】
DEVMODE 構造体や DEVNAMES 構造体のハンドルを解放します。

【関数書式】
(1)C++Builder
BOOL IKReleaseDevModeHandleEx(HANDLE hDevMode, HANDLE hDevNames);
(2)Delphi
function IKReleaseDevModeHandleEx(hDevMode, hDevNames: THandle): LongBool;

【引数】

名称	内容
hDevMode	**IKGetDevModeHandleEx** や **IKPrintDialog** で取得した DEVMODE 構造体のハンドル
hDevNames	**IKGetDevModeHandleEx** や **IKPrintDialog** で取得した DEVNAMES 構造体のハンドル

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
DEVMODE 構造体や DEVNAMES 構造体のハンドルが不要になった段階で実行します。
成功した場合、hDevMode や hDevNames が占有するメモリは解放されます。
hDevNames が NULL または 0 の場合、IKReleaseDevModeHandle と同じ動作となります。

IKRoundRect

【機能】
描画先オブジェクトに角が丸の矩形を描画します。

【関数書式】
(1)C++Builder
BOOL IKRoundRect(LPVOID DeviceValue, LPRECT PrintRect, int x, int y, PTR_IKPRINT_DRAWINFO DrawInfo, BYTE DeviceMode);
(2)Delphi
function IKRoundRect(DeviceValue: THandle; var PrintRect: TRect; x, y: Integer; var DrawInfo: IKPRINT_DRAWINFO; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
PrintRect	矩形を指定する構造体変数
x,y	丸い角を描画するための楕円の幅と高さ
DrawInfo	描画する情報を指定する構造体変数
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PrintRect で指定した座標を基に矩形を描画し、その角が指定の楕円のパラメータによって丸められます。アウトラインは **PenStyle** の値で描画され、内部は **BrushStyle** の値で塗りつぶされます。(描画対象はスクリーン、プリンタ、メモリハンドル)
描画するには DrawInfo の **PenWidth**,**PenStyle**,**PenMode**,**PenColor**,**BrushStyle**,**BrushColor**,**Transparent**,**BackColor** に値を設定する必要があります。**BackColor** は **Transparent** = 0 でペンが実線以外、あるいはブラシがハッチパターンの場合に有効です。

PrintRect のメンバー変数と x,y の値をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
PrintRect のメンバー変数と x,y の値を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5RoundRect:	DeviceValue, PrintRect, x, y, PrintInfo
IKRoundRect:	DeviceValue, PrintRect, x, y, DrawInfo, DeviceMode

PrintInfo と DrawInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。Transparent を 0 以外に設定し、IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

【ImageKit6 との違い】
IKPRINT_DRAWINFO 構造体に BackColor が追加されました。Transparent が 0 以外であれば IK6 と同じ動作です。

IKSaveDevModeHandle

【機能】
DEVMODE 構造体のハンドルをファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKSaveDevModeHandle(LPCTSTR PrintFileName, HANDLE hDevMode);
(2)Delphi
function IKSaveDevModeHandle(PrintFileName: PChar; hDevMode: THandle): LongBool;

【引数】

名称	内容
PrintFileName	保存するプリンタ設定ファイル名
hDevMode	**IKGetDevModeHandle(Ex)**や **IKPrintDialog** で取得した DEVMODE 構造体のハンドル

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PrintFileName に保存されるファイル形式は **IKSetPrint** で保存されるファイル形式と同じです。
そのため、保存したファイルは IKGetDevModeHandle(Ex),IKPrintCreateDC(Ex),IKSetPrint で使用可能です。
DEVMODE 構造体については WindowsAPI 関連の書籍などをご覧ください。

IKSaveDevModeHandleEx

【機能】
DEVMODE 構造体と DEVNAMES 構造体のハンドルをファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKSaveDevModeHandleEx(LPCTSTR PrintFileName, HANDLE hDevMode, HANDLE hDevNames);
(2)Delphi
function IKSaveDevModeHandleEx(PrintFileName: PChar; hDevMode, hDevNames: THandle): LongBool;

【引数】

名称	内容
PrintFileName	保存するプリンタ設定ファイル名
hDevMode	**IKGetDevModeHandle(Ex)**や **IKPrintDialog** で取得した DEVMODE 構造体のハンドル
hDevNames	**IKGetDevModeHandleEx** や **IKPrintDialog** で取得した DEVNAMES 構造体のハンドル

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
PrintFileName に保存されるファイル形式は **IKSetPrint** で保存されるファイル形式と同じです。
そのため、保存したファイルは IKGetDevModeHandle(Ex),IKPrintCreateDC(Ex),IKSetPrint で使用可能です。
hDevNames が NULL または 0 の場合、IKSaveDevModeHandle と同じ動作となります。
DEVMODE 構造体と DEVNAMES 構造体については WindowsAPI 関連の書籍などをご覧ください。

IKSetDefaultPrinter

【機能】
通常使うプリンタ(デフォルトプリンタ)を設定します。

【関数書式】
(1)C++Builder
BOOL IKSetDefaultPrinter(LPCTSTR PrinterName);
(2)Delphi
function IKSetDefaultPrinter(PrinterName: PChar): LongBool;

【引数】

名称	内容
PrinterName	設定するプリンタ名

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
Windows のプリンタフォルダに登録されているプリンタを通常使うプリンタとして設定できます。
PrinterName には IKEnumPrinters 関数で取得したリストに含まれるプリンタ名を設定します。

IKSetDevModeInfo

【機能】
DEVMODE 構造体のハンドルが指す内容を更新します。

【関数書式】
(1)C++Builder
BOOL IKSetDevModeInfo(HANDLE hDevMode, PTR_IKPRINT_DEVMODEINFO DevModeInfo);
(2)Delphi
function IKSetDevModeInfo(hDevMode: THandle; var DevModeInfo: IKPRINT_DEVMODEINFO): LongBool;

【引数】

名称	内容
hDevMode	**IKGetDevModeHandle(Ex)**で取得した DEVMODE 構造体のハンドル
DevModeInfo	印刷情報を設定する構造体(ユーザ定義型)変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
DevModeInfo で設定された情報を DEVMODE 構造体のハンドルに反映させます。
DevModeInfo の各メンバー変数に 0 を設定した場合は、何の変更も行いません。(Collate は除く)
ただし、反映させた結果についてはプリンタドライバに依存するため、その後で印刷を行っても効果がまったくなかったり、異なる設定を行っても印刷結果が同じになることがあります。詳しくはプリンタドライバのマニュアルなどを参照してください。
DEVMODE 構造体については WindowsAPI 関連の書籍などをご覧ください。
IKPRINT_DEVMODEINFO については、**「Ik10Print.dll/Ik10PrintA.dll/Ik10Print64.dll/Ik10Print64A.dll」の構造体の定義**をご覧ください。

コード例:
(1)C++Builder

```
HANDLE hDevMode;
IKPRINT_DEVMODEINFO DevModeInfo;
BOOL Ret;
HDC hDC;

hDevMode = IKGetDevModeHandle("EPSON LP-8200C", NULL);
if (!hDevMode) return;

Ret = IKGetDevModeInfo(hDevMode, &DevModeInfo);
if (DevModeInfo.Copies != 3)
  DevModeInfo.Copies = 3; //印刷部数を 3 に
Ret = IKSetDevModeInfo(hDevMode, &DevModeInfo);

hDC = IKPrintCreateDC(NULL, NULL, hDevMode, 2);
if (hDC)
{
  //印刷処理・・・
  Ret = IKPrintDeleteDC(hDC);
}

Ret = IKReleaseDevModeHandle(hDevMode);
```

(2)Delphi

```
hDevMode: THandle;
DevModeInfo: IKPRINT_DEVMODEINFO;
Ret: LongBool;
```

```
DC: HDC;

hDevMode := IKGetDevModeHandle('EPSON LP-8200C', nil);
if hDevMode = 0 then Exit;

Ret := IKGetDevModeInfo(hDevMode, DevModeInfo);
if DevModeInfo.Copies <> 3 then
   DevModeInfo.Copies := 3; //印刷部数を 3 に
Ret := IKSetDevModeInfo(hDevMode, DevModeInfo);

DC := IKPrintCreateDC(nil, nil, hDevMode, 2);
if DC <> 0 then
begin
   //印刷処理・・・
   Ret := IKPrintDeleteDC(DC);
end;

Ret := IKReleaseDevModeHandle(hDevMode);
```

IKSetPixel

【機能】
描画先オブジェクトの指定したピクセルに指定したカラーを設定します。

【関数書式】
(1)C++Builder
BOOL IKSetPixel(LPVOID DeviceValue, int x, int y, COLORREF Color);
(2)Delphi
function IKSetPixel(DeviceValue: THandle; x, y: Integer; Color:COLORREF): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル（DeviceMode による）
x,y	ピクセルの座標
Color	ピクセルの新しいカラー
DeviceMode	描画対象（0:スクリーン、2:メモリハンドル）

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
x,y で指定したピクセルに Color で指定したカラーを設定します。
指定のカラーをデバイスが正確に再現できない場合には、近似のカラーが用いられます。
（対象はスクリーン、メモリハンドル）
DeviceMode が 0 もしくは 2 の場合に有効になります。x,y の値はピクセル単位で設定してください。
Color には **RGB**(Red,Green,Blue)として求めた値などを設定してください。

【ImageKit5 との違い】

関数名	**引数の並び**
IK5SetPixel:	DeviceValue, x, y, Color
IKSetPixel:	DeviceValue, x, y, Color, DeviceMode

IK5SetDeviceMode で設定した値を引数の DeviceMode に渡すと IK5 と同じ動作となります。

IKSetPrint

【機能】
プリンタ設定ダイアログで設定した情報をファイルに保存します。

【関数書式】
(1)C++Builder
BOOL IKSetPrint(HWND hWnd, LPCTSTR PrintFileName);
(2)Delphi
function IKSetPrint(hWnd: HWND; PrintFileName: PChar): LongBool;

【引数】

名称	内容
hWnd	ウィンドウハンドル
PrintFileName	プリンタ設定ファイル名

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
プリンタ設定ダイアログを表示して、設定した情報をファイルに保存します。その際に引数 PrintFileName+".PrnTxt"という名称のファイルに、テキストとしてプリンタ名・出力先名を保存します。内訳は 1 レコード目がプリンタ名、2 レコード目が出力先名となります。レコードのデリミタ(区切り)は{CR}{LF}です。
(プリンタ設定ダイアログを終了する際に OK を選択した場合、上記の 2 つのファイルを保存し、戻り値 0 以外を返します。)
保存した設定ファイルは、IKGetDevModeHandle(Ex),IKPrintCreateDC(Ex)で使用します。(テキストとして保存したファイルは使用しません。)
PrintFileName にファイル名のみを設定すると、プリンタ設定ファイルはカレントフォルダに保存されます。

IKTextOut

【機能】
描画先オブジェクトにテキストを描画します。

【関数書式】
(1)C++Builder
BOOL IKTextOut(LPVOID DeviceValue, int x, int y, PTR_IKPRINT_TEXTINFO TextInfo, LPCTSTR Text, BYTE DeviceMode);
(2)Delphi
function IKTextOut(DeviceValue: THandle; x, y: Integer; var TextInfo: IKPRINT_TEXTINFO; Text: PChar; DeviceMode: Byte): LongBool;

【引数】

名称	内容
DeviceValue	デバイスコンテキストもしくはラスタイメージのメモリハンドル(DeviceMode による)
x,y	描画開始位置の座標
TextInfo	描画する情報を指定する構造体変数
Text	描画する文字列(終端がヌル)
DeviceMode	描画対象(0:スクリーン、1:プリンタ、2:メモリハンドル)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
x,y で指定した座標からテキストを描画します。(対象はスクリーン、プリンタ、メモリハンドル)
描画するには TextInfo の **CharSet**,**TextColor1**,**TextColor2**,**FontName**,**FontSize**,**Transparent**,**Direction**,**CharAngle**, **RotateString** に値を設定する必要があります。**CharExtra** に 1 以上の値を設定すると文字の間隔を考慮します。
RotateString が 0 の場合、**CharAngle** は 0,90,180,270 以外は無効です。文字列の方向に関わらず、x,y から常に右方向に描画されます。**RotateString** が 0 以外の場合は、**Direction** は無効です。x,y を始点として文字列を回転させた方向に描画されます。Text に改行コードを含む複数行の文字列を設定しても当関数では正しく出力されないため、単一行に切り出して複数回出力を行うか、IKDrawText 関数をご使用ください。
TextInfo の CharExtra と x,y の値をピクセル単位として扱う場合
DeviceMode が 0 もしくは 2
TextInfo の CharExtra と x,y の値を 0.1mm 単位として扱う場合
DeviceMode が 1

【ImageKit5 との違い】

関数名	**引数の並び**
IK5TextOut(A):	DeviceValue, x, y, PrintInfo
IKTextOut:	DeviceValue, x, y, TextInfo, Text, DeviceMode

PrintInfo と TextInfo は構造体の構造が異なりますが、使用する際に必要なメンバー変数はほぼ同じです。
IK5PRINT_DATA --> IKPRINT_TEXTINFO
CharExtra,CharSet,RotateString, Alpha1,Alpha2,HotkeyPrefix が新たに追加され、Text が構造体から削除されました。
TextForeColor,TextBackColor が TextColor1,TextColor2 にそれぞれ変更されました。
IK5PRINT_DATA_A --> IKPRINT_TEXTINFO
CharSet,RotateString,Alpha1,Alpha2,HotkeyPrefix が新たに追加され、Text が構造体から削除されました。
TextForeColor,TextBackColor が TextColor1,TextColor2 にそれぞれ変更されました。
PrintInfo.Text を引数の Text に、IK5SetDeviceMode で設定した値を引数の DeviceMode に、TextInfo の CharSet を-1 に、RotateString を 0 に設定し、TextForeColor,TextBackColor を TextColor1,TextColor2 にそれぞれ変更すると IK5 と同じ動作となります。

【ImageKit6 との違い】
IKPRINT_TEXTINFO 構造体に Alpha1,Alpha2,HotkeyPrefix が追加され、TextForeColor,TextBackColor が TextColor1,TextColor2 にそれぞれ変更されました。TextForeColor,TextBackColor を TextColor1,TextColor2 にそれぞれ変更すると IK6 と同じ動作となります。

１－５. Ik10RasToVect.dll/Ik10RasToVectA.dll/Ik10RasToVect64.dll/Ik10RasToVect64A.dll

ラスタイメージをベクトルイメージに変換する機能を提供します。

●DLL 関数コマンド一覧（アルファベット順）

関数名	内容
IKRasterToVector	ラスタイメージをベクトルイメージに変換

【ユーザ関数の定義】

【関数書式】
(1)C++Builder　BOOL _stdcall UserProc(short Percent);
(2)Delphi　function UserProc(Percent: Smallint): LongBool; stdcall;

【引数】

名称	内容
Percent	現在処理している%数

【戻り値】
False(0)の場合は実行している処理を終了します。True(0 以外)は処理を継続します。

ユーザ関数は UserProc という名称で説明しておりますが、実際の名称は何を設定されても構いません。関数名が UserProc の場合は以下のような形式で引数に渡します。
(1)C++Builde　UserProc
(2)Delphi　LONG_PTR(Addr(UserProc)) or LONG_PTR(@UserProc)

IKRasterToVector

【機能】
ラスタイメージをベクトルイメージに変換します。

【関数書式】
(1)C++Builder
HANDLE IKRasterToVector(HANDLE Handle, BOOL BlackOnWhite, short Tolerance, long TimeOutSeconds, IKPROCESSPROC UserProc, LPCTSTR Caption, LPCTSTR Message, LPCTSTR Button);
(2)Delphi
function IKRasterToVector(Handle: THandle; BlackOnWhite: LongBool; Tolerance: Smallint; TimeOutSeconds, UserProc: LONG_PTR; Caption, Message, Button: PChar): THandle;

【引数】

名称	内容
Handle	ラスタイメージのメモリハンドル
BlackOnWhite	False(0):黒地に白字、True(0 以外):白地に黒字
Tolerance	許容値 0(感度高)～10(感度低)
TimeOutSeconds	処理を中止するタイムアウト(秒)
UserProc	ユーザ関数のアドレス(ユーザ関数を使用しない場合は 0 を指定します。)
Caption	処理中のダイアログボックスのタイトルバーに表示する文字列
Message	処理中のダイアログボックスの中央に表示する文字列
Button	処理中のダイアログボックスのボタンに表示する文字列

【戻り値】
ベクトルイメージのメモリハンドル(実行に失敗した場合は 0 が返されます)。

【解説】
白黒 2 値イメージが対象です。
処理中に TimeOutSeconds で設定された時間を超えた場合、強制的に処理を中断します。
ユーザ関数が設定されている場合や Caption、Message、Button が空の文字列の場合は、処理中のダイアログボックスは表示されません。ダイアログボックスが表示される場合は、処理進捗状況がゲージ上に%形式で表示されます。ユーザ関数については「Ik10RasToVect.dll/Ik10RasToVectA.dll/Ik10RasToVect64.dll/Ik10RasToVect64A.dll」のユーザ関数の定義をご覧ください。

1－6. Ik10Scan.dll/Ik10ScanA.dll/Ik10Scan64.dll/Ik10Scan64A.dll

TWAIN データソースから、イメージを取り込む機能などを提供します。

●DLL 関数コマンド一覧（アルファベット順）

関数名	内容
IKScanCloseDS	データソースのクローズ
IKScanExec	データソースからの画像の取り込み
IKScanFreeTwain	TWAIN DLL の解放
IKScanGetBitDepth	データソースがサポートしている画素ビット数の取得
IKScanGetCapEnum	指定した項目の値の列挙
IKScanGetCapRange	指定した項目の範囲の取得
IKScanGetDSInfo	指定したデータソースの情報を取得
IKScanGetMinimumSize(Ex)	データソースから取り込める最小サイズの取得
IKScanGetPhysicalSize(Ex)	データソースから取り込める最大物理サイズの取得
IKScanGetPixelType	データソースがサポートしているピクセルタイプの取得
IKScanInitialize	TWAIN の初期化処理
IKScanIsCapSupported	指定した項目をサポートしているかどうかの判定
IKScanList	インストールされているデータソースの列挙
IKScanListLen	インストールされているデータソースの列挙文字列の長さを取得
IKScanLoadTwain	TWAIN DLL のロード
IKScanOpenDS	データソースのオープン
IKScanSelect	データソースの選択
IKScanTerminate	TWAIN の終了処理

●取り込みの簡単な流れ

[IKScanLoadTwain]
　　|
IKScanInitialize（必須）
　　|
IKScanGetDSInfo, IKScanList, IKScanListLen, IKScanSelect
　　|
[IKScanOpenDS]
　　|
IKScanGetBitDepth, IKScanGetCapEnum, IKScanGetCapRange, IKScanGetPhysicalSize(Ex)
IKScanGetPixelType, IKScanGetMinimumSize(Ex), IKScanIsCapSupported
　　|
IKScanExec
　　|
[IKScanCloseDS]
　　|
IKScanTerminate（必須）
　　|
[IKScanFreeTwain]

注意：

[ ]の IKScanLoadTwain/FreeTwain と IKScanOpenDS/CloseDS は実行しなくても構いませんが、実行する時は IKScanLoadTwain/FreeTwain と IKScanOpenDS/CloseDS を対で使用してください。IKScanLoadTwain/Twain と IKScanInitialize/Terminate を除く全ての関数は、その間であればどのタイミングでも使用可能です。
ただし、IKScanOpenDS/CloseDS を実行する場合、IKScanOpenDS/CloseDS の間に記載されている関数は IKScanOpenDS を実行した後に使用しなければなりません。

●構造体（ユーザ定義型）の定義および解説

IKSCAN_DATASOURCEINFO:　　TWAIN データソースの情報を参照。

(1)C++Builder

```
typedef struct {
    WORD        ProtocolMajor;
    WORD        ProtocolMinor;
    char        Manufacturer[34];
    WORD        ManufacturerLen;
    char        ProductFamily[34];
    WORD        ProductFamilyLen;
    char        ProductName[34];
    WORD        ProductNameLen;
    WORD        SourceMajor;
    WORD        SourceMinor;
    char        SourceVersionInfo[34];
    WORD        SourceVersionInfoLen;
} IKSCAN_DATASOURCEINFO;
typedef IKSCAN_DATASOURCEINFO * PTR_IKSCAN_DATASOURCEINFO;
```

(2)Delphi

```
type
  IKSCAN_DATASOURCEINFO = Record
    ProtocolMajor:      Word;
    ProtocolMinor:      Word;
    Manufacturer:       array[0..33] of AnsiChar;
    ManufacturerLen:    Word;
    ProductFamily:      array[0..33] of AnsiChar;
    ProductFamilyLen:   Word;
    ProductName:        array[0..33] of AnsiChar;
    ProductNameLen:     Word;
    SourceMajor:        Word;
    SourceMinor:        Word;
    SourceVersionInfo:  array[0..33] of AnsiChar;
    SourceVersionInfoLen:Word;
  end;
```

ProtocolMajor: データソースがサポートする TWAIN のメジャーバージョン番号。
ProtocolMinor: データソースがサポートする TWAIN のマイナーバージョン番号。
Manufacturer: 製造者名。
ManufacturerLen: Manufacturer 文字列の長さ。 *1
ProductFamily: 製品ファミリー名。
ProductFamilyLen: ProductFamily 文字列の長さ。 *1
ProductName: 製品名。
ProductNameLen: ProductName 文字列の長さ。 *1
SourceMajor: データソースのメジャーバージョン番号。
SourceMinor: データソースのマイナーバージョン番号。
SourceVersionInfo: データソースのバージョン情報
SourceVersionInfoLen: SourceVersionInfoLen 文字列の長さ。 *1

*1 4 バイト型から 2 バイト型に変更されました。ImageKit6 から移行する場合は注意してください。

IKSCAN_EXEC: 画像を取得する際に設定します。

(1)C++Builder

```
typedef struct {
    WORD        seSize;
    WORD        PixelType;
    float       Left;
    float       Top;
    float       Right;
    float       Bottom;
    long        XResolution;
```

```
    long        YResolution;
    float       XScaling;
    float       YScaling;
    WORD        BitDepth;
    short       Brightness;
    short       Contrast;
    float       Gamma;
    short       Highlight;
    short       Threshold;
    short       Shadow;
    short       Indicator;
    short       ScanMode;
    LPSTR       ScanDsName;
    WORD        UiMode;
    WORD        UnitMode;
    WORD        UnitFlag;
    short       PageCount;
    WORD        Orientation;
    WORD        TransferMode;
    WORD        Compression;
    WORD        PaperSize;
    WORD        BitDepthReduction;
    LPSTR       HalfTone;
    WORD        FileFormat;
    short       JpegQuality;
    LPSTR       FileName;
    WORD        ImageFilter;
    WORD        NoiseFilter;
    WORD        DropoutColor;
    WORD        BorderDetection;
    WORD        Deskew;
    WORD        ScanningSpeed;
    WORD        MoireFilter;
    WORD        Sharpness;
    WORD        RotateBack;
    WORD        ExtUiMode;
    WORD        DynamicThreshold;
    float       ColorBWRatio;
    WORD        IgnoreBackColor;
    WORD        OverScan;
    long        SkipBlankPage;
    float       SkipBlankThreshold;
    WORD        RemoveHole;
    WORD        AdjustGamma;
    short       FocusPosition;
    long        Rotation;
    LPTSTR      InformationFileName;
    short       TextEnhancement;
    WORD        ImageMerge;
    WORD        Border;
    short       BorderColor;
    WORD        AutoBright;
} IKSCAN_EXEC;
typedef IKSCAN_EXEC * PTR_IKSCAN_EXEC;
```

(2)Delphi

```
type
  IKSCAN_EXEC = Record
    seSize:                 Word;
```

```
    PixelType:          Word;
    Left:               Single;
    Top:                Single;
    Right:              Single;
    Bottom:             Single;
    XResolution:        Longint;
    YResolution:        Longint;
    XScaling:           Single;
    YScaling:           Single;
    BitDepth:           Word;
    Brightness:         Smallint;
    Contrast:           Smallint;
    Gamma               Single;
    Highlight:          Smallint;
    Threshold:          Smallint;
    Shadow:             Smallint;
    Indicator:          WordBool;
    ScanMode:           Smallint;
    ScanDsName:         PAnsiChar;
    UiMode:             Word;
    UnitMode:           Word;
    UnitFlag:           Word;
    PageCount:          Smallint;
    Orientation:        Word;
    TransferMode:       Word;
    Compression:        Word;
    PaperSize:          Word;
    BitDepthReduction:  Word;
    HalfTone:           PAnsiChar;
    FileFormat:         Word;
    JpegQuality:        Smallint;
    FileName:           PAnsiChar
    ImageFilter:        Word;
    NoiseFilter:        Word;
    DropoutColor:       Word;
    BorderDetection:    Word;
    Deskew:             Word;
    ScanningSpeed:      Word;
    MoireFilter:        Word;
    Sharpness:          Word;
    RotateBack:         Word;
    ExtUiMode:          Word;
    DynamicThreshold:   Word;
    ColorBWRatio:       Single;
    IgnoreBackColor:    Word;
    OverScan:           Word;
    SkipBlankPage:      Longint;
    SkipBlankThreshold: Single;
    RemoveHole:         Word;
    AdjustGamma:        Word;
    FocusPosition:      Smallint;
    Rotation:           Longint;
    InformationFileName: PChar;
    TextEnhancement:    Smallint;
    ImageMerge:         Word;
    Border:             Word;
    BorderColor:        Smallint;
    AutoBright:         Word;
end;
```

seSize: IKSCAN_EXEC 構造体のサイズを設定します。 *1
C++Builder では sizeof 演算子、Delphi では SizeOf 関数などを使用して値を設定します。0 を設定した場合は、AutoBright までのメンバを参照します。

PixelType: 取り込むイメージの種類を設定します。
0:白黒 2 値(1 ビット)
1:グレースケール(4 ビット、8 ビット、12 ビット、14 ビット、16 ビット)
2:RGB カラー(24 ビット、36 ビット、42 ビット、48 ビット)
3:パレットカラー(4 ビット、8 ビット)
1000:白黒/カラー自動判別(1 ビット or 24 ビット)
1001:白黒 2 値とグレースケール(1 ビットと 8 ビット)
1002:白黒 2 値と RGB カラー(1 ビットと 24 ビット)
UiMode が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合は無効です。
データソースがサポートしているピクセルタイプは、IKScanGetCapEnum で取得できます。
1000 はパナソニック製スキャナ用ドライバ(メモリ転送時は非圧縮形式のみ)、キヤノン製 DR スキャナ用ドライバ、エプソン製スキャナ用ドライバ、PFU 製スキャナ用ドライバご利用時に有効です。
1001,1002 はキヤノン製 DR スキャナ用ドライバ、パナソニック製スキャナ用ドライバ(メモリ転送時は非圧縮形式のみ)と PFU 製スキャナ用ドライバ、エプソン製スキャナ用ドライバご利用時に有効です。1000,1001,1002 がサポートされているかどうかについては IKScanIsCapSupported で判定できます。

Left、Top、Right、Bottom: 取り込み範囲を **UnitMode** に応じた値で設定します。
(例えば、0 の場合はインチ単位、1 の場合は cm 単位、5 の場合はピクセル単位)
データソースのデフォルト値を使用する場合は Left、Top、Right、Bottom に 0 を設定します。
取り込み終了後に **UnitMode** に応じた値が返されます。**UiMode** が 2(UI 非表示)以外の場合や **PaperSize** が 1 以上およびデジカメからの取り込みの場合、設定値は無効です。

XResolution、YResolution: X 方向および Y 方向の DPI(解像度)を設定します。
データソースのデフォルト値を使用する場合は-1 を指定します。
UiMode が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合は無効です。データソースがサポートしている解像度の範囲や値は、IKScanGetCapEnum もしくは IKScanGetCapRange で取得できます。
(注意)
X 方向と Y 方向の読み取り解像度には異なる値を設定できますが、データソースによっては異なる解像度をサポートしていないものがあります。その場合、X 方向および Y 方向のどちらかの解像度が有効になります。実際の読み取り解像度は BeforeScanProc 関数の引数を参照するか、IKScanExec 終了後に当メンバー変数を参照してください。

XScaling、YScaling: X 方向および Y 方向のスケーリングの比率を設定します。(デフォルトは 1.0)
等倍(実寸)で読み取る場合は 1.0 です。**UiMode** が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合、およびデータソースが機能をサポートしていない場合は無効です。データソースがサポートしているスケーリングの比率の範囲や値は、IKScanGetCapEnum もしくは IKScanGetCapRange で取得できます。
(注意)
X 方向と Y 方向のスケーリングの比率には異なる値を設定できますが、データソースによっては異なるスケーリングをサポートしていないものがあります。その場合、X 方向および Y 方向のどちらかのスケーリングが有効になります。

BitDepth: 画素ビット数(1,4,8,12,14,16,24,36,42,48)を設定します。
12,14,16 もしくは 36,42,48 を設定して取り込みを行う場合は、**TranferMode** を 2 に設定してください。データソースでサポートしている値にも関わらずエラーとなる場合は、0 を設定してください。0 の場合は **PixelType** に応じたデフォルト値を使用します。**UiMode** が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合は無効です。

Brightness: 明るさ(-1000～1000、デフォルトは 0)を設定します。
Contrast: コントラスト(-1000～1000、デフォルトは 0)を設定します。
キヤノン製 DR スキャナ用ドライバご利用時で PixelType が 0 で BitDepthReduction が 0、TextEnhancement が 1 以上の場合は、1～255(デフォルトは 128)の範囲が有効です。
PFU 製スキャナ用ドライバご利用時で PixelType が 1000,1001,1002 の場合は、-127～127(デフォルトは 0)の範囲が有効です。
PixelType が 0 の場合、**BitDepthReduction** の値により効果がない場合があります。(下記は効果がある場合)

PixelType	0	1,2,3

Brightness	BitDepthReduction = 0(*),1	○
Contrast	BitDepthReduction = 1	○

(*)しきい値の機能を明るさで表すデータソースで使用(Threshold をサポートしていないデータソースなど)

UiMode が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合、およびデータソースが機能をサポートしていない場合は無効です。データソースがサポートしている明るさ・コントラストの範囲や値は、IKScanGetCapEnum もしくは IKScanGetCapRange で取得できます。

Gamma: ガンマ(デフォルトは 2.2)を設定します。
UiMode が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合、およびデータソースが機能をサポートしていない場合は無効です。データソースがサポートしているガンマの範囲や値は、IKScanGetCapEnum もしくは IKScanGetCapRange で取得できます。

Highlight: ハイライト(0～255、デフォルトは 255)を設定します。
PixelType が 0 以外の場合に有効です。UiMode が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合、およびデータソースが機能をサポートしていない場合は無効です。データソースがサポートしているハイライトの範囲や値は、IKScanGetCapEnum もしくは IKScanGetCapRange で取得できます。

Threshold: しきい値(0～255,デフォルトは 128)を設定します。
PixelType が 0、BitDepthReduction が 0 の場合に有効ですが、しきい値の機能を明るさで表すデータソースの場合は Threshold ではなく Brightness をご使用ください。UiMode が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合、およびデータソースが機能をサポートしていない場合は無効です。データソースがサポートしているしきい値の範囲や値は、IKScanGetCapEnum もしくは IKScanGetCapRange で取得できます。

Shadow: シャドウ(0～255、デフォルト 0)を設定します。
PixelType が 0 以外の場合に有効です。UiMode が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合、およびデータソースが機能をサポートしていない場合は無効です。データソースがサポートしているシャドウの範囲や値は、IKScanGetCapEnum もしくは IKScanGetCapRange で取得できます。

Indicator: 取り込みインジケーターの有無を設定します。
0:非表示、0 以外:表示
UiMode が 2(UI 非表示)以外の場合およびデータソースが機能をサポートしていない場合は無効です。

ScanMode: 取り込みモードを設定します。
スキャナ(0:反射原稿-原稿台、1:反射原稿-ADF 片面、2:反射原稿-ADF 両面、7:透過原稿-ポジフィルム、フィルムホルダ使用、8:透過原稿-ポジフィルム、フィルムエリアガイド使用)
デジカメ(3:オリジナル画像、4:オリジナル画像で配列を使用、5:サムネイル画像、6:サムネイル画像で配列を使用)
メーカ提供 UI の設定情報ファイルを使用(100)
UiMode が 2(UI 非表示)以外の場合は無効です。
ScanMode が 4,6 の場合は、IKScanExec 関数の引数で与える配列が有効となりますが、それ以外は無効です。
ImageKit5 と互換性を保つため、XResolution = 0, ScanMode = 1 の場合はデジカメからの取り込みとなります。(取り込む画像は主画像となります。)
7,8 の場合はエプソン製スキャナ用ドライバご利用時に有効です。8 はフィルムエリアガイドをサポートする機種で使用可能です。
100 の場合は InformationFileName にメーカ提供 UI の設定情報が保存されたファイル名を設定します。
(注意)
両面読み取り機能をサポートしているスキャナでも、ユーザインタフェースを非表示にすると正しく読み取りできない場合があります。

ScanDsName: 取り込みを行うデータソースの名称を設定します。
""や NULL を設定した場合、IKScanSelect で選択されたデータソースとなります。
IKScanOpenDS を使用した場合は無効です。

UiMode: ユーザインタフェースの設定を行います。
0:UI 表示-取込後閉じない、1:UI 表示-取込後閉じる、2:UI 非表示、3:UI 表示－値の設定のみで取り込み不可
UI 非表示での取り込みや UI を表示して値の設定のみできる機能をサポートしているかどうかについては、IKScanIsCapSupported で判定できます。

UnitMode: 取り込み時の単位を設定します。

0:インチ、1:cm(センチメートル)、2:パイカ、3:ポイント、4:twip、5:ピクセル、6:mm(ミリメートル)
UnitFlag が 0 の場合は、データソースがサポートしている値でなければなりません。
UiMode が 2(UI 非表示)以外の場合、設定値は無効ですが、取り込み時に使用した単位を返します。データソースがサポートしている取り込み単位は、IKScanGetCapEnum で取得できます。

UnitFlag:
取り込み時の単位をデータソースの機能に任せるか、ImageKit10 で考慮するかを設定します。(0:データソースの機能を利用 1:ImageKit10 で考慮)
例えば、データソースがサポートしていない単位を使用したい場合や、**PaperSize** が 0 以下で **Left**、**Top**、**Right**、**Bottom** を設定しているのに正しく取り込みができない、または取り込み範囲が有効にならない場合などは 1 を指定してください。
XResolution、**YResolution** が共に-1 で **UnitMode** が 5 の場合は無効です。
UiMode が 2(UI 非表示)以外の場合は無効です。

PageCount:
取り込みを行うページ数(デジカメの場合は画像数)を設定します。ADF にある全てのページ、およびデジカメ内の全ての画像を取り込む場合は 0 以下を指定します。デジカメからの取り込みで配列を使用する場合は、その要素数を指定します。**ScanMode** が 2 の場合は奇数を設定しても取り込まれるページ数は偶数になります(両面取り込みのため)。
UiMode が 2(UI 非表示)以外の場合や原稿台からの取り込みの場合は無効です。
PixelType が 1001,1002 の場合は 1 ページにつき 2 つのイメージが作成されるため、偶数を指定してください。たとえば、1 ページの原稿を両面で 1001,1002 で取り込む場合は 0 以下または 4 を指定します。なお、パナソニック製スキャナ用ドライバご利用時に 1001,1002 で取り込む場合は常に 0 以下として取り扱います(1 以上は無視されます)。
(注意)
データソースが機能をサポートしていない場合や正常に機能しない場合は、**PageCount** 分取り込みを行った後で処理を中断いたしますので、機種によっては ADF から用紙が取り込まれた状態のまま、用紙が内部に残る場合があります。

Orientation: 用紙の向き、または画像の回転角度を設定します。
0:0 度(ポートレイト)、1:90 度、2:180 度、3:270 度(ランドスケープ)、1000:文字向き検知または自動回転
用紙の向きは 0 度を基準に時計回りに回転します。**UiMode** が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合は無効です。原稿台や ADF に置いた用紙の出力結果を回転させたり、原稿台や ADF に用紙を横向きにセットする場合などに使用します。データソースがサポートしている用紙の向きは、IKScanGetCapEnum で取得できます。
1000 は読み取った画像の中にある文字の向きを認識し、文字の向きが正常になるように画像を 90 度単位で回転させます。キヤノン製 DR スキャナ用ドライバでは「文字向き検知」、パナソニック製スキャナ用ドライバでは「原稿方向補正」という表現を用いています。
また、TWAIN 仕様書に記載されている自動回転に対応しているドライバについても 1000 にて動作いたします。1000 がサポートされているかどうかについては IKScanIsCapSupported で判定できます。1000 をサポートしているデータソースで 1000 を指定した出力結果が適切でない場合は用紙を縦向きにセットしてお試しください。

TransferMode: イメージの転送方法を設定します。
0:ネイティブ(DIB で転送)、1:ファイル、2:メモリ(ブロック単位で転送)、3:ファイル 2(*)
*ImageKit10 ではサポートしておりません。
データソースがサポートしている転送方法は、IKScanGetCapEnum で取得できます。

Compression: ファイルおよびメモリ転送時(**TransferMode**=1 or 2)の圧縮方法を設定します。
0:非圧縮、1:PACKBITS、2:GROUP3-1D、3:GROUP3-1DEOL、4:GROUP3-2D、5:GROUP4、6:JPEG、7:LZW、
8:JBIG、9:PNG、10:RLE4、11: RLE8、12:BITFIELDS (2,3,4,5 は FAX[CCITT]、10,11,12 は BMP 形式)
開発者作成 UI で取り込む場合は、**PixelType**、**BitDepth** に適切な値を設定してください。。たとえば、GROUP3/4 圧縮の場合は白黒 2 値、JPEG 圧縮の場合は 8 ビットグレースケールまたは 24 ビットカラーでなければなりません。
データソースがサポートしている圧縮方法は、IKScanGetCapEnum で取得できます。
データソースによってはファイル転送時の圧縮方法は非圧縮のみ有効となる場合があります。

PaperSize: 取り込む用紙サイズを設定します。
0:ユーザ定義サイズ、1:A4、2:JIS B5、3:US レター、4:US リーガル、5:A5、6:ISO B4、7:ISO B6、9:US レジャー
10:US エグゼクティブ、11:A3、12:ISO B3、13:A6、14:C4、15:C5、16:C6、17:4A0、18:2A0、19:A0、20:A1、21:A2
22:A7、23:A8、24:A9、25:A10、26:ISO B0、27:ISO B1、28:ISO B2、29:ISO B5、30:ISO B7、31:ISO B8
32:ISO B9、33:ISO B10、34:JIS B0、35:JIS B1、36:JIS B2、37:JIS B3、38:JIS B4、39:JIS B6、40:JIS B7、41:JIS B8
42:JIS B9、43:JIS B10、44:C0、45:C1、46:C2、47:C3、48:C7、49:C8、50:C9、51:C10、52:US ステイトメント
53:ビジネスカード、54:最大サイズ、1000:未定義サイズ

Left、**Top**、**Right**、**Bottom** を有効にする場合は 0 以下を指定します。

UiMode が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合は無効です。
また、データソースによってはユーザ定義サイズ(0)と読み取り範囲を設定しても、開始位置などが正しく反映されない場合があります。その場合は、定型サイズ(1 以上)を符号反転してお試しください。
C++Builder と Delphi は符号なし型のため、マイナス値をキャストして設定してください。
例: 用紙サイズを符号反転し、読み取り位置を設定

(1)C++Builder

```
IKSCAN_EXEC sc;

sc.PaperSize = (WORD)-3;
sc.RectLeft = 0;
sc.RectTop = 0;
sc.RectRight = 6.9;
sc.RectBottom = 10.2;
```

(2)Delphi

```
sc: IKSCAN_EXEC;

sc.PaperSize := Word(-3);
sc.RectLeft := 0;
sc.RectTop := 0;
sc.RectRight := 6.9;
sc.RectBottom := 10.2;
```

PaperSize にはデータソースがサポートしている用紙サイズ、もしくはその中で一番大きな用紙サイズを設定してください。
なお、データソースによっては効果がない場合もありますので、詳しくはスキャナメーカにご確認ください。

(注意)
データソースがサポートしている用紙サイズは IKScanGetCapEnum で取得できます。データソースによっては定型サイズをサポートしていないものや、用紙サイズの機能自体をサポートしていないものがありますので、その場合は 0 を指定してください。
未定義サイズ(1000)を設定すると、用紙サイズを自動検知して取り込みを行いますが、用紙の幅と高さのどちらか一方のみを検知するスキャナでは用紙の自動検知が正しく行われない場合があります。
未定義サイズがサポートされているかどうかについては IKScanIsCapSupported で判定できます。
未定義サイズがサポートされていて PaperSize に 1000 を設定しても用紙サイズの自動検知が正しく動作しない場合は次の手順をお試しください。
(1)BorderDetection に 0 以外を設定する。この段階で正しく動作せずに PFU 製スキャナ用ドライバをご利用の場合は(2)へ。
(2)PaperSize に 1000 に加えて読み取り可能な最大サイズの用紙サイズを設定する。たとえば、最大サイズが A3 であれば PaperSize = 1000 + 11 を設定する。

BitDepthReduction: 色深度の減少方法を設定します。(PixelType=0 の場合に有効です)
0:しきい値、1:ハーフトーン、2:カスタムハーフトーン(*)、3:拡散、4:動的しきい値
*ImageKit10 ではサポートしておりません。
0 の場合は Threshold(データソースによっては Brightness)が有効となり、1 の場合は HalfTone,Brightness,Contrast が有効となります。UiMode が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合は無効です。

データソースがサポートしている値は、IKScanGetCapEnum で取得できます。仮にデータソースが BitDepthReduction の機能をサポートしていなくても、Threshold もしくは HalfTone のどちらかをサポートしている場合は、0 もしくは 1 を設定してください。

HalfTone: ハーフトーンの名称を設定します。
PixelType=0、BitDepthReduction=1 の場合に使用します。データソースがサポートしているハーフトーンの名称は、IKScanGetCapEnum で取得できます。UiMode が 2(UI 非表示)以外の場合やデジカメからの取り込みの場合は無効です。

FileFormat: ファイル転送時(TransferMode=1)のファイルフォーマットを設定します。 *1
0:TIFF、1:PICT、2:BMP、3:XBM、4:JFIF(JPEG)、5:FPX、6:TIFF MULTI、7:PNG、8:SPIFF、9:EXIF、10:PDF
11:JP2(JPEG2000 Part1)、13:JPX(JPEG2000 Part2)、14:DEJAVU、15:PDF/A(Version 1)、16:PDF/A(Version 2)
フォーマットに応じて Compression にも適切な値を設定します。
データソースがサポートしている圧縮方法の値は、IKScanGetCapEnum で取得できます。

JpegQuality： *1
ファイルおよびメモリ転送（TransferMode=1 or 2）で圧縮方法が JPEG（Compression=6）の場合に設定する品質係数を示します。
-4：未定義、-3:低品質、-2：中品質、-1:高品質、0 ～ 100
データソースがサポートしている品質係数の範囲や値は、IKScanGetCapEnum もしくは IKScanGetCapRange で取得できます。

FileName： *1
ファイル転送時（TransferMode=1）およびメモリ転送の圧縮モード時（TransferMode=2,Compression が 0 以外）に読み込んだ画像を保存するファイル名を設定します（255 バイト以下）。空文字列および NULL,nil を設定するとファイルは作成されません。ファイル転送時にはファイル名は必須です。
UiMode が 2 で ScanMode が 0 の場合は設定されたファイル名で保存しますが、それ以外の場合は拡張子の前に 4 桁の連番を挿入します。（例: Image.bmp → Image0001.bmp）

ImageFilter： イメージフィルタを設定します。 *1
0：なし、1:自動、2：LOWPASS、3:BANDPASS、4: HIGHPASS
画像の品質を改善します。LOWPASS はハーフトーン画像、BANDPASS はテキストが含まれる画像、HIGHPASS は線が含まれる画像に適しています。データソースがサポートしているイメージフィルタは、IKScanGetCapEnum で取得できます。
UiMode が 2(UI 非表示)以外の場合およびデータソースが機能をサポートしていない場合は無効です。

NoiseFilter： ノイズフィルタを設定します。 *1
0：なし、1:自動、2：LONEPIXEL、3:MAJORITYRULE
白黒 2 値画像からノイズを除去します。データソースがサポートしているノイズフィルタは、IKScanGetCapEnum で取得できます。UiMode が 2(UI 非表示)以外の場合およびデータソースが機能をサポートしていない場合は無効です。

DropoutColor： ドロップアウトカラーを設定します。 *1
0：赤、1:緑、2：青、3:なし、4：白
PixelType=0,1 の場合に有効です。ドロップアウトカラーを無効にする場合は、3 もしくは 4 を設定します。データソースがサポートしているノイズフィルタは、IKScanGetCapEnum で取得できます。UiMode が 2(UI 非表示)以外の場合およびデータソースが機能をサポートしていない場合は無効です。

BorderDetection： 自動領域切り出しの有無を設定します。 *1
0:無効、0 以外:有効
用紙サイズに応じて領域を切り出します。PaperSize が 1000 以上の時に有効な機能です。UiMode が 2(UI 非表示)以外の場合およびデータソースが機能をサポートしていない場合は無効です。データソースが機能をサポートしているかどうかについては IKScanIsCapSupported で判定できます。

Deskew： デスキューの有無を設定します。 *1
0:無効、1:有効、2：有効（滑らか）
原稿の傾きを補正します。基本的には PaperSize が 1000 以上の時に有効な機能ですが、データソースによっては定型サイズでも有効な場合があります。滑らか設定はパナソニック製スキャナ用ドライバご利用時のみ有効です（メモリ転送時は非圧縮形式のみ）。UiMode が 2(UI 非表示)以外の場合およびデータソースが機能をサポートしていない場合は無効です。データソースが機能をサポートしているかどうかについては IKScanIsCapSupported で判定できます。

ScanningSpeed： 読み取り速度を設定します。 *1
エプソン製スキャナ用ドライバおよびパナソニック製スキャナ用ドライバご利用時に有効です。
(1)エプソン製スキャナ用ドライバ　　0：品質優先、1:速度優先
(2)パナソニック製スキャナ用ドライバ 0：低速、1:通常、2：高速
読み取り速度を速くすると遅い場合に比べて画質が粗くなりますので、用途に応じてご利用ください。
データソースがサポートしている読み取り速度は、IKScanGetCapEnum で取得できます。UiMode が 2(UI 非表示)以外の場合およびデータソースが機能をサポートしていない場合は無効です。

MoireFilter： モアレ除去フィルタのレベルを設定します。 *1
エプソン製スキャナ用ドライバ、キヤノン製 DR スキャナ用ドライバ、PFU 製スキャナ用ドライバ、パナソニック製スキャナ用ドライバご利用時に有効です。エプソン製スキャナ用ドライバご利用時に段階設定が可能なのはエプソンスキャナドライバ『EPSON Scan』ご利用時となります。
(1)エプソン製スキャナ用ドライバ
0：なし、1:標準（汎用的な設定）、2:85lpi（新聞など）、3:133lpi（週刊誌やカタログなど）、4：175lpi（写真など）

(2)キヤノン製 DR スキャナ用ドライバ
1:なし、2:高速、3:高画質
(3)PFU 製スキャナ用ドライバ
0:なし、1:モアレ除去レベル 1、2:モアレ除去レベル 2、3:モアレ除去レベル 3、4:モアレ除去レベル 4
(4)パナソニック製スキャナ用ドライバ
0:なし、1:あり

キヤノン製 DR スキャナ用ドライバでは解像度が 300dpi 以下の場合に有効です。PFU 製スキャナ用ドライバでは **Sharpness** が 0 の場合に有効です。パナソニック製スキャナ用ドライバではモアレ除去とマルチストリームを同時に使用することはできません(どちらか一方を無効にしなければなりません)。
データソースがサポートしているモアレ除去フィルタは、IKScanGetCapEnum で取得できます。なお、エプソン製スキャナ用ドライバのバージョンによっては有効無効の 2 つしか設定できないものがありますが、ImageKit ではその場合 0 か 0 以外で判定します。**UiMode** が 2(UI 非表示)以外の場合およびデータソースが機能をサポートしていない場合は無効です。

Sharpness: シャープネスのレベルを設定します。 *1
キヤノン製 DR スキャナ用ドライバ、エプソン製スキャナ用ドライバ、パナソニック製スキャナ用ドライバ、PFU 製スキャナ用ドライバご利用時に有効です。
エプソン製スキャナ用ドライバご利用時に段階設定が可能なのはエプソンスキャナドライバ『EPSON Scan』ご利用時となります。
キヤノン製 DR スキャナ用ドライバでは「エッジ強調」、エプソン製スキャナ用ドライバでは「アンシャープマスク」、パナソニック製スキャナ用ドライバでは「画質」、PFU 製スキャナ用ドライバでは「輪郭強調」という表現を用いています。
(1)キヤノン製 DR スキャナ用ドライバ 0～5
0 の場合はドライバのデフォルト値 3 と同じ動作となります。1 の場合は画像の輪郭が柔らかくなり、5 の場合は画像の輪郭がくっきりします。
(2)エプソン製スキャナ用ドライバ 0:なし、1:弱、2:中、3:強
(3)パナソニック製スキャナ用ドライバ 0:なし、1:スムーズ、2:低、3:標準、4:高、5:自動
(4)PFU 製スキャナ用ドライバ 0:なし、1:弱、2:中、3:強

データソースがサポートしているシャープネスは、IKScanGetCapEnum で取得できます。なお、エプソン製スキャナ用ドライバのバージョンによっては有効無効の 2 つしか設定できないものがありますが、ImageKit ではその場合 0 か 0 以外で判定します。**UiMode** が 2(UI 非表示)以外の場合およびデータソースが機能をサポートしていない場合は無効です。

RotateBack: ADF を使用した両面スキャン時の裏面回転を設定します。 *1
キヤノン製 DR スキャナ用ドライバおよびエプソン製スキャナ用ドライバ『EPSON Scan』ご利用時に有効ですが、『EPSON Scan』の Ver.1.x においては A4 サイズを超える原稿では無効です。
0:裏面を回転しない、1:裏面を 180 度回転する
値が 1 の場合、キヤノン製 DR スキャナ用ドライバでは UI を表示した時の「裏面を+180 度回転する」のオプションを有効にした場合と同等です。エプソン製スキャナ用ドライバでは『EPSON Scan』の環境設定において「裏面の向きを表面に合わせる」オプションを有効にした場合と同等です。データソースが機能をサポートしているかどうかについては IKScanIsCapSupported で判定できます。**UiMode** が 2(UI 非表示)で **ScanMode** が 2(ADF 両面)の場合に有効です。

ExtUiMode: 拡張 UI モードを設定します。 *1
エプソン製スキャナ用ドライバご利用時に有効です。
0:前回設定の UI モードで起動、1:プロフェッショナルモードで起動、2:全自動モードで起動、3:ホームモードで起動
4:全自動モードで起動(エラーメッセージを除き、UI 非表示)、5:オフィスモードで起動
7:シートフィード専用モードで起動
各設定値は『EPSON Scan』の UI で標準で表示されるもののみ設定可能です。
(例:ES シリーズで 2,4、GT シリーズで ADF 非接続時 5／ADF 接続時 2,4 は設定不可)
UiMode が 0,1(UI 表示)の場合に有効です。データソースがサポートしている値は、IKScanGetCapEnum で取得できます。

DynamicThreshold: ダイナミックスレッシュホールドのレベルを設定します。 *1
パナソニック製スキャナ用ドライバご利用時に有効です。
0:なし、1:明るい、2:少し明るい、3:標準、4:少し暗い、5:暗い
ダイナミックスレッシュホールド機能を有効にすると、文字や絵がつぶれないよう、かすれないよう、最適な読み取りを行います。
データソースがサポートしている値は、IKScanGetCapEnum で取得できます。**UiMode** が 2(UI 非表示)で **PixelType** が 0 の場合に有効です。**DynamicThreshold** に 1 以上を設定すると、**BitDepthReduction**,**HalfTone**,**Sharpness**,**Threshold** の値は無効となります。

ColorBWRatio: 白黒原稿の中に含まれるカラー画素の割合を設定します。 *1
パナソニック製スキャナ用ドライバご利用時に有効です。
0.01 から 50.00 までの値を%単位で設定します。値が小ければ白黒原稿はカラーとして認識される確率が高くなり、大きければ白黒として認識される確率が高くなります。UiMode が 2(UI 非表示)で PixelType が 1000 の場合に有効です。

IgnoreBackColor: 白黒/カラー自動判別時に原稿の背景色を無視するかどうかを設定します。 *1
パナソニック製スキャナ用ドライバご利用時に有効です。
0:背景色を無視しない、1:背景色を無視する
UiMode が 2(UI 非表示)で PixelType が 1000 の場合に有効です。

OverScan: オーバースキャンの取り扱いを設定します。 *2
0:指定されたサイズで読み取りを行う、1:指定されたサイズを考慮して自動で読み取りを行う
2:指定されたサイズを基に上下を広げて読み取りを行う、3:指定されたサイズを基に左右を広げて読み取りを行う
4:指定されたサイズを基に上下左右を広げて読み取りを行う
傾いた用紙を取り込む場合などに取り込み領域(Left,Top,Right,Bottom,PaperSize)を広げることができます。
UiMode が 2(UI 非表示)の場合に有効です。ただし、デジタルカメラからの取り込みやデータソースが機能をサポートしていない場合は無効です。データソースがサポートしている値は IKScanGetCapEnum で取得できます。

SkipBlankPage: 白紙ページの取り扱いを設定します。 *2
-2:白紙ページを除去しない、-1:白紙ページを除去する
キヤノン製 DR スキャナ用ドライバ
0:白紙ページを除去しない、1:白紙ページを除去する
エプソン製スキャナ用ドライバ
0:白紙ページを除去しない、1:白紙ページを除去する(レベル低)、2:白紙ページを除去する(レベル中)
3:白紙ページを除去する(レベル高)
白紙ページを除去します。キヤノン製 DR スキャナ用ドライバとパナソニック製スキャナ用ドライバでは
SkipBlankThreshold も併せて設定します。
キヤノン製 DR スキャナ用ドライバで白紙ページを除去する場合は-1 または 1 を設定してください。また、ScanMode を必ず 1 に設定してください。白紙ページ除去が有効の場合は ScanMode が 1 であっても両面取り込みとなります。
エプソン製スキャナ用ドライバで白紙ページを除去する場合は-1 または 1 以上を設定してください。白紙ページの判別レベルを設定する場合は 1 以上とし、ガンマ補正(AdjustGamma)がサポートされている場合は AdjustGamma を 1 に設定してください。AdjustGamma が 0 の場合、白紙ページ除去機能が正常に動作しない可能性があります。また、ガンマ補正(AdjustGamma)がサポートされていない場合は、デフォルトの設定でスキャンを行うと暗めの画像が取得され、白紙ページ除去機能が正常に動作しない可能性があります。その場合は Brightness や Gamma の値を調整して明るめの画像にしてください。
UiMode が 2(UI 非表示)の場合に有効です。ただし、デジタルカメラからの取り込みやデータソースが機能をサポートしていない場合は無効です。
データソースがサポートしている値は IKScanGetCapEnum もしくは IKScanGetCapRange で取得できます。

SkipBlankThreshold: 白紙ページに含まれるしきい値の割合や白紙スキップのしやすさを設定します。 *2
キヤノン製 DR スキャナ用ドライバ
0.1 から 20.0 までの値を%単位で設定します。
パナソニック製スキャナ用ドライバ
0.01 から 5.00 までの値を%単位で設定します。
デフォルトは 1 です。キヤノン製 DR スキャナ用ドライバとパナソニック製スキャナ用ドライバご利用時に有効です。キヤノン製 DR スキャナ用ドライバでは白紙スキップのしやすさを表し、値が大きければ白紙ページとして認識される確率が高くなります。パナソニック製スキャナ用ドライバでは白紙ページに含まれるしきい値の割合を表し、値が小ければ白紙ページとして認識される確率が高くなります。
UiMode が 2(UI 非表示)、SkipBlankPage が-1 または 1 の場合に有効です。

RemoveHole: パンチ穴の取り扱いを設定します。 *2
キヤノン製 DR スキャナ用ドライバ、エプソン製スキャナ用ドライバ、パナソニック製スキャナ用ドライバ、PFU 製スキャナ用ドライバご利用時に有効です。
0:パンチ穴を除去しない、1:パンチ穴を除去する(白で埋める)、2:パンチ穴を除去する(周辺色で埋める)
デフォルトは 0 です。"白で埋める"と"周辺色で埋める"の機能は PFU 製スキャナ用ドライバご利用時のみ有効です。
UiMode が 2(UI 非表示)の場合に有効です。
ただし、デジタルカメラからの取り込みやデータソースが機能をサポートしていない場合は無効です。
データソースが機能をサポートしているかどうかについては、IKScanIsIsCapSupported で判定できます。

AdjustGamma: ガンマ補正係数を設定します。 *2
エプソン製スキャナ用ドライバご利用時に有効です。
0:係数 1.0、1:係数 1.8
UiMode が 2(UI 非表示)の場合に有効です。
データソースが機能をサポートしているかどうかについては、IKScanIsCapSupported で判定できます。

FocusPosition: 焦点位置を設定します。 *2
エプソン製スキャナ用ドライバご利用時に有効です。
デフォルトは 0 です。焦点位置(原稿にピントが合うセンサの位置)を調整できます。
焦点位置は 0.1mm 単位で設定します。たとえば、1.0mm の場合は 10 を設定します。
UiMode が 2(UI 非表示)の場合に有効です。
データソースがサポートしている焦点位置の範囲や値は、IKSacanGetCapRange で取得できます。

Rotation: 画像の回転角度を設定します。 *2
-360～360(単位は度)。デフォルトは 0 です。
0 度を基準に負の値の場合は反時計回り、正の値の場合は時計回りに回転します。UiMode が 2(UI 非表示)の場合に有効です。Orientation と機能が似ていますが、Orientation への設定可能な値が 0 のみの場合は Rotaion を使用してください。Orientation と Rotaion の機能を両方ともサポートするデータソースの場合、双方の値によって回転角度が決まったり、Rotaion の値が有効になったりします。そのため、Rotaion に 0 以外を設定する場合は Orientation に 0 を設定するといいでしょう。
ただし、Orientation に 1000 が設定されている場合やデジタルカメラからの取り込み、およびデータソースが機能をサポートしていない場合は無効です。
データソースがサポートしている値は IKScanGetCapEnum もしくは IKScanGetCapRange で取得できます。

InformationFileName: メーカ提供 UI の設定情報を保存するファイル名を設定します。 *2
メーカ提供 UI を表示して設定した情報をファルに保存すると、そのファイルを開発者作成 UI にて使用することができます。一度メーカ提供 UI を表示する必要はありますが、設定情報を保存すると、次回から開発者作成 UI にて取り込みが可能です。

TextEnhancement: テキストエンハンスメントの種類を設定します。 *2
キヤノン製 DR スキャナ用ドライバ
0:適用しない、1:テキストエンハンスメント、2:アドバンスドテキストエンハンスメント、3:高速テキストエンハンスメント
4,5:アドバンスドテキストエンハンスメントⅡ(ご利用の機種により値が異なる)
エプソン製スキャナ用ドライバ
0:適用しない、1:文字くっきり(標準)、2:文字くっきり(強)
キヤノン製 DR スキャナ用ドライバ、エプソン製スキャナ用ドライバご利用時に有効です。エプソン製スキャナ用ドライバご利用時に段階設定が可能なのはエプソンスキャナドライバ『EPSON Scan Ver.5.0 以降』ご利用時となります。
キヤノン製 DR スキャナ用ドライバでは PixelType が 0 で BitDepthReduction が 0 の場合に有効です。
キヤノン製 DR スキャナ用ドライバでは「テキストエンハンスメント」、エプソン製スキャナ用ドライバでは「文字くっきり」という表現を用いています。
デフォルトは 0 です。UiMode が 2(UI 非表示)の場合に有効です。
データソースがサポートしている値は、IKScanGetCapEnum で取得できます。
なお、エプソン製スキャナ用ドライバのバージョンによっては有効無効の 2 つしか設定できないものがありますが、ImageKit ではその場合 0 か 0 以外で判定します。データソースが機能をサポートしていない場合は無効です。

ImageMerge: 両面合成モードを設定します。 *3
0:なし(合成しない)、1:表面が上で裏面が下になる(上下の貼り合わせ)、2:裏面が上で表面が下になる(上下の貼り合わせ)、3:表面が左で裏面が右になる(左右の貼り合わせ)、4:裏面が左で表面が右になる(左右の貼り合わせ)
ADF 両面取り込み時に 0 以外を設定すると表面と裏面を合成して、一つの画像データにします。両面合成機能を使用して得られる出力イメージは、回転を組み合わせることによっても変わります。デフォルトは 0 です。UiMode が 2、ScanMode が 2 の場合に有効です。
データソースがサポートしている値は、IKScanGetCapEnum で取得できます。

Border: 境界補正を行うかどうかを設定します。 *3
エプソン製スキャナ用ドライバ
0:境界補正しない、1:境界補正する
パナソニック製スキャナ用ドライバ
0:境界削除しない、1:境界削除する - 自動モード(指定された幅まで原稿領域の縁を検出して補正)

2:境界削除する - 固定幅モード(指定された幅で画像の縁から一律の領域を補正)
エプソン製スキャナ用ドライバ、パナソニック製スキャナ用ドライバご利用時に有効です。エプソン製スキャナ用ドライバでは「境界補正」(原稿端付近に発生する背景色の写りこみや影の補正、または原稿端への枠の付加)、パナソニック製スキャナ用ドライバでは「境界削除」(読み取りやコピーにおいて発生する画像周辺の黒い縁を取り除く)という表現を用いています。
デフォルトは 0 です。**UiMode** が 2 の場合に有効です。1 以上の場合、**BorderColor** に設定した色で塗りつぶしが行われます。データソースが機能をサポートしているかどうかについては、IKScanIsIsCapSupported で判定できます。

BorderColor: 境界補正色を設定します。 *3
エプソン製スキャナ用ドライバ
0:白色、1:黒色
パナソニック製スキャナ用ドライバ
0:周辺色(原稿端の色)、1:原稿背景色、2:白色
エプソン製スキャナ用ドライバ、パナソニック製スキャナ用ドライバご利用時に有効です。
デフォルトは 0 です。**UiMode** が 2、**Border** が 1 以上の場合に有効です。補正の適用範囲はエプソン製スキャナ用ドライバ、パナソニック製スキャナ用ドライバともユーザインタフェース(メーカ提供 UI)で設定された値が使用されます。

AutoBright: 取り込む時の明るさを自動で処理するかどうかを設定します。 *4
0:明るさを手動で処理する、0 以外:明るさを自動で処理する
デフォルトは 0 です。**UiMode** が 2 の場合に有効です。データソースが機能をサポートしているかどうかについては、IKScanIsIsCapSupported や IKScanGetCapEnum で判定できます。データソースが TWAIN2.0 以降に対応している場合は、IKScanGetCapEnum でサポートする値を取得できます。
0 またはデータソースが機能をサポートしてない場合は、Brightness の値が有効になります。

IKSCAN_EXEC は、ImageKit5 の IK5SCANE_EXEC の後継にあたりますが、次の変更点があります。
1.Pixeltype を PixelType に
2.Resolution を XResolution、YResolution に分割
3.Indicator を 4 バイト型から 2 バイト型に
4.InUnitMode,OutUnitMode を UnitMode に統一
5.いくつかのメンバー変数を新たに追加
(XScaling,YScaling,UnitFlag,PageCount,Orientation,TransferMode,Compression,PaperSize,BitDepthReduction, HalfTone)

*1 ImageKit7 で追加されたメンバー変数です。ImageKit5/6 から移行する場合は注意してください。
*2 ImageKit8 で追加されたメンバー変数です。ImageKit5/6/7 から移行する場合は注意してください。
*3 ImageKit10 で追加されたメンバー変数です。ImageKit5/6/7/8 から移行する場合は注意してください。
*4 ImageKit10 で追加されたメンバー変数です。ImageKit5/6/7/8/9 から移行する場合は注意してください。

IKSCAN_IMAGEINFO: 取り込む画像の情報を取得。

(1)C++Builder
```
typedef struct {
    float       XResolution;
    float       YResolution;
    long        Width;
    long        Height;
    short       BitDepth;
    short       PixelType;
    WORD        Compression;
} IKSCAN_IMAGEINFO;
typedef IKSCAN_IMAGEINFO * PTR_IKSCAN_IMAGEINFO;
```

(2)Delphi
```
type
  IKSCAN_IMAGEINFO = Record
    XResolution:        Single;
    YResolution:        Single;
    Width:              Longint;
    Height:             Longint;
```

```
    BitDepth:       Smallint;
    PixelType:      Smallint;
    Compression:    Word;
  end;
```

XResolution: X 方向の解像度。 *1
YResolution: Y 方向の解像度。 *1
Width: 転送される画像の幅。(ピクセル)
Height: 転送される画像の高さ。(ピクセル)
BitDepth: 画素ビット数。 *2
PixelType: ピクセルタイプ。(0～3)
Compression: 圧縮方法。 *3

*1 TWAIN の仕様では取り込み単位(インチや cm など)に依存した値になりますが、データソースによっては取り込み単位に関わらず、常に DPI となるものがあります。また、ピクセル単位で取り込むと 1 が設定される場合があります。

*2 12,14 ビットグレースケールは 16 となり、36,42 ビットカラーは 48 となります。
*3 ファイル転送とメモリ転送の際に使用されます。(ネイティブ転送では意味を持ちません。)

IKSCAN_RANGE: 項目の範囲を取得。

(1)C++Builder

```
typedef struct {
    float       Min;
    float       Max;
    float       Step;
    float       Default;
    float       Current;
} IKSCAN_RANGE;
typedef IKSCAN_RANGE * PTR_IKSCAN_RANGE;
```

(2)Delphi

```
type
  IKSCAN_RANGE = Record
    Min:        Single;
    Max:        Single;
    Step:       Single;
    Default:    Single;
    Current:    Single;
  end;
```

Min: 最小値。
Max: 最大値。
Step: ステップ値。 *1
Default: デフォルト値。 *2
Current: 現在設定されている値。

*1 範囲内(Min～Max)の隣接する値との差。
*2 デバイスの電源を入れた時の値。

IKScanCloseDS

【機能】
データソースをクローズします。

【関数書式】
(1)C++Builder
BOOL IKScanCloseDS(HANDLE AppHandle, HANDLE SrcHandle);
(2)Delphi
function IKScanCloseDS(AppHandle, SrcHandle: THandle): LongBool;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル
SrcHandle	**IKScanOpenDS** で取得したデータソースハンドル

【戻り値】
成功の場合は True(0 以外)、失敗の場合は False(0)を返します。

【解説】
オープンしたデータソースをクローズすると共に、IKScanOpenDS で取得したデータソースハンドルを解放します。SrcHandle に無効な値を設定した場合はエラーとなります。

IKScanExec

【機能】
TWAIN データソースからイメージの取込を実行します。

【関数書式】
(1)C++Builder
int IKScanExec(HANDLE AppHandle, HANDLE SrcHandle, PTR_IKSCAN_EXEC Image, IK_BEFORESCANPROC BeforeScanProc, IK_SCANNINGPROC ScanningProc, IK_AFTERSCANPROC AfterScanProc, DWORD *Index);
(2)Delphi
function IKScanExec(AppHandle, SrcHandle: THandle; var Image: IKSCAN_EXEC; BeforeScanProc, ScanningProc, AfterScanProc: Longint; var Index: DWORD): Integer;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル
SrcHandle	**IKScanOpenDS** で取得したデータソースハンドルもしくは 0
Image	取込条件を設定する構造体(ユーザ定義型)変数
BeforeScanProc	イメージ取り込み前に処理するユーザ関数のアドレス(必要ない場合は 0 を指定)
ScanningProc	イメージ取り込み中に処理するユーザ関数のアドレス(必要ない場合は 0 を指定)
AfterScanProc	イメージ取り込み後に処理するユーザ関数のアドレス(必要ない場合は 0 を指定) ユーザ関数名が AfterScanProc の場合は以下のような形式で引数に渡します。 (1)C++Builder　AfterScanProc (2)Delphi　LONG_PTR(Addr(AfterScanProc)) or LONG_PTR(@AfterScanProc)
Index	デジカメから取り込む際に使用するインデックス配列 (1)C++Builder　配列の先頭のポインタを渡す (2)Delphi　Ix: array [0..2] of DWORD の場合は、Ix[0]を与える ※インデックス配列を使用しない場合は NULL もしくはダミー配列を与えてください。

【戻り値】
取込んだイメージの数(取込んだイメージがない場合は 0 が返されます)。

【解説】
SrcHandle に IKScanOpenDS で取得した値を使用すると、オープン中のデータソースは他のアプリケーションからは利用できなくなり、データソースを占有することができます。
Image は、取込条件用の構造体(ユーザ定義型)変数です。IKSCAN_EXEC については、
「Ik10Scan.dll/Ik10ScanA.dll/Ik10Scan64.dll/Ik10Scan64A.dll」の構造体の定義をご覧ください。
SrcHandle が 0 の場合 ScanDSName は必須ですが、0 以外の場合は無効です。
Index は UiMode が 2 で ScanMode がデジカメを表す場合に有効となりますが、Index が必要ない場合は NULL もしくはダミー配列を渡してください。
Index は、デジタルカメラから指定した画像を取り込む際に使用します。(ScanMode が 4,6 の時に有効)
例えば、5 番目の画像と 8 番目の画像を取り込むには
Index[0] = 5
Index[1] = 8
とします。
その際、配列の要素数 2 を PageCount に設定します。

イメージを 1 枚取り込む前後および最中に、ユーザ関数(**BeforeScanProc**,**AfterScanProc**,**ScanningProc**)に制御が移りますので、その中で処理したいコードを記述してください。データソースによってはサポートしていない機能があります。その場合はサポートされている機能のみを使用して取り込みを行います。

＜関数を実行する前に設定する必要のある IKSCAN_EXEC 構造体のメンバ変数＞
「必須項目」
ScanDsName(空文字列を指定する場合は、予め **IKScanSelect** 関数を実行しておくこと)
※**IKScanOpenDS** 関数を事前に実行した場合は **ScanDsName** は無効(オープンされたデータソースが有効)

TransferMode,Compression,UiMode(エプソン製スキャナ用ドライバご利用時は ExtUiMode も必須)
「設定条件により必要な項目」
FileFormat,FileName,InformationFileName,JpegQuality

UiMode が 2 で ScanMode が 100 の場合
ScanMode,PageCount,UnitMode

UiMode が 2 で ScanMode が 100 以外の場合
(1)スキャナ
AdjustGamma,AutoBright,BitDepth,BitDepthReduction,Border,BorderColor,BorderDetection,Brightness,ColorBWRatio,Contrast,Deskew,DropoutColor,DynamicThreshold,FocusPosition,Gamma,HalfTone,Highlight,IgnoreBackColor,ImageFilter,ImageMerge,Indicator,MoireFilter,NoiseFilter,Orientation,OverScan,PageCount,PaperSize,PixelType,RemoveHole,Rotation,ScanMode,ScanningSpeed,Shadow,Sharpness,SkipBlankPage,SkipBlankThreshold,TextEnhancement,Threshold,UnitFlag,UnitMode,XResolution,XScaling,YResolution,YScaling
PaperSize<=0: Bottom,Left,Right,Top
※
AdjustGamma,Border,BorderColor,ColorBWRatio,DynamicThreshold,FocusPosition,IgnoreBackColor,MoireFilter,RemoveHole,RotateBack,ScanningSpeed,Sharpness,SkipBlankThreshold,TextEnhancement はご利用のドライバによっては無効です。

(2)デジタルカメラ
PageCount,ScanMode,UnitMode　　※ScanMode=1 の場合は、XResolution = 0 とすること。

＜関数実行後に取得される IKSCAN_EXEC 構造体のメンバ変数＞
BitDepth,Bottom,Compression,FileFormat(ファイル転送の場合),Left,PixelType,Right,Top,UnitMode,
XResolution,YResolution

(開発者作成 UI を使用する上での注意事項)
UiMode が 2 の場合、機種やプロパティの設定状態により読み取りに失敗する場合があります。エラーが発生する主な原因を次に示しますので参考にしてください。

・設定した取り込み単位がサポートされていない。サポートされている値は IKScanGetCapEnum 関数で取得できます。
・設定したピクセルタイプがサポートされていない。サポートされている値は IKScanGetCapEnum もしくは IKScanGetCapPixelType 関数で取得できます。
・設定した画素ビット数がサポートされていない。サポートされている値は IKScanGetBitDepth 関数で取得できます。
・設定した解像度がサポートされていない。サポートされている値は IKScanGetCapEnum もしくは IKScanGetCapRange 関数で取得できます。
・設定した用紙サイズがサポートされていない。サポートされている値は IKScanGetCapEnum 関数で取得できます。
(1)用紙サイズそのものがサポートされていない場合
PaperSize に 0(ユーザ定義サイズ)を設定し、Left,Top,Right,Bottom に範囲内の値を設定する。読み取り範囲は、IKScanGetMinimumSize(Ex)と IKScanGetPhysicalSize(Ex)関数で取得できます。
(2)設定した用紙サイズがサポートされているのに読み取りに失敗する場合
PaperSize に 0(ユーザ定義サイズ)以外を設定してご確認ください。

【ImageKit5 との違い】

関数名	引数の並び
IK5ScanExec:	hWnd, Image, ScanUserProc
IKScanExec:	AppHandle, SrcHandle, Image, BeforeScanProc, ScanningProc, AfterScanProc, Index

IK6 から hWnd を IKScanInitialize で渡し、その戻り値の AppHandle を使用します。
IK5 の ScanUserProc は IK6 以降の AfterScanProc にあたります。Image の構造は IK5 と IK10 で異なります。

【ImageKit6 との違い】
IKSCAN_EXEC 構造体にメンバー変数が追加されました。

【ImageKit7 との違い】
IKSCAN_EXEC 構造体にメンバー変数が追加されました。

【ImageKit8 との違い】

IKSCAN_EXEC 構造体にメンバー変数が追加されました。

【ImageKit9 との違い】
IKSCAN_EXEC 構造体にメンバー変数が追加されました。

【ユーザ関数の定義】

BeforeScanProc: イメージを取り込む前に制御が移ります。
【関数書式】
(1)C++Builder
BOOL _stdcall BeforeScanProc(PTR_IKSCAN_IMAGEINFO ImageInfo, long Count);
(2)Delphi
function BeforeScanProc(var ImageInfo: IKSCAN_IMAGEINFO; Count: Integer): LongBool; stdcall;

【引数】

名称	内容
ImageInfo	取り込もうとするイメージの情報
Count	取り込もうとするイメージの順番(1～)

【戻り値】
False(0)の場合は取り込みを終了します。True(0 以外)の場合は処理を継続します。

IKSCAN_EXEC 構造体の **PaperSize** に 1000 以上を設定すると ImageInfo の Width と Height が-1 になる場合があります。また、実際の原稿サイズよりも Width,Height とも大きくなる場合があります。
IKSCAN_EXEC 構造体の **PixelType** に 1000 を設定すると ImageInfo の BitDepth と PixelType が実際に取り込んだ画像と異なる場合があります。

ユーザ関数は BeforeScanProc という名称で説明しておりますが、実際の名称は何を設定されても構いません。

ScanningProc: イメージを取り込んでいる最中に制御が移ります。
【関数書式】
(1)C++Builder
BOOL _stdcall ScanningProc(short Percent);
(2)Delphi
function ScanningProc(Percent: Smallint): LongBool; stdcall;

【引数】

名称	内容
Percent	取り込んでいるイメージの進捗状況(0～100、パーセントで示す)

【戻り値】
False(0)の場合は取り込みを終了します。True(0 以外)の場合は処理を継続します。

この関数はメモリ転送のみが対象で、ネイティブ転送やファイル転送では無効です。また、メモリ転送でも圧縮形式の場合と **BeforeScanProc** 関数の引数 ImageInfo の Width と Height が-1 の場合は無効です。
ユーザ関数は ScanningProc という名称で説明しておりますが、実際の名称は何を設定されても構いません。

AfterScanProc: イメージを取り込んだ後に制御が移ります。
【関数書式】
(1)C++Builder
BOOL _stdcall AfterScanProc(HANDLE DibHandle, HANDLE OrgHandle, long Count, short BitOrder);
(2)Delphi
function AfterScanProc(DibHandle, OrgHandle: THandle; Count: Integer; BitOrder): LongBool; stdcall;

【引数】

名称	内容

DibHandle	Windows で処理できる DIB のメモリハンドル
OrgHandle	Windows で処理できないメモリハンドル （12,14,16 ビットグレースケールや 36,42,48 ビットカラー、および圧縮モード時）
Count	取り込んだイメージの数
BitOrder	取り込んだイメージのビットがバイトの左端から始まっているか、右端から始まっているかを表します。 0：右端から（LSB）、1：左端から(MSB)

【戻り値】

False(0)の場合は取り込みを終了します。True(0 以外)の場合は処理を継続します。

※AfterScanProc の後で取り込んだイメージのメモリハンドルを解放しています（DibHandle と OrgHandle の両方）。
メモリハンドルを残しておきたい場合はメモリのコピーを実行してください。

(1)ネイティブ転送、メモリ転送で非圧縮モードの場合
取り込んだイメージが 1,4,8,24 ビットイメージの場合は、イメージを DibHandle に設定し OrgHandle は 0 になります。
取り込んだイメージが 12,14,16 ビットグレースケールもしくは 36,42,48 ビットカラーの場合は、イメージを OrgHandle に設定し、OrgHandle を 8 ビットグレースケールもしくは 24 ビットカラーに減色したイメージを DibHandle に設定します。ただし、DibHandle 用のメモリが確保できない場合は 0 となります。

(2)メモリ転送で圧縮モードの場合
DibHandle は 0 で OrgHandle に取り込んだイメージを設定します。

(3)ファイル転送の場合
DibHandle,OrgHandle とも 0 になります。

IKSCAN_EXEC 構造体の UiMode を 2（UI 非表示）、かつ UnitMode を 5（ピクセル）に設定すると、DibHandle と OrgHandle の 2 つのメモリハンドルの解像度情報が 1 になる場合があります。

ユーザ関数は AfterScanProc という名称で説明しておりますが、実際の名称は何を設定されても構いません。

【ImageKit5 との違い】

IK6 から引数に BitOrder が追加されています。

IKScanFreeTwain

【機能】
IKScanLoadTwain でロードした TWAIN DLL を解放します。

【関数書式】
(1)C++Builder
BOOL IKScanFreeTwain(void);
(2)Delphi
function IKScanFreeTwain(): LongBool;

【引数】
ありません。

【戻り値】
成功の場合は True(0 以外)、失敗の場合は False(0)を返します。

【解説】
IKScanLoadTwain を実行していない場合は IKScanFreeTwain を実行する必要はありません（ImageKit 内で自動的にロード/解放が行われます）。IKScanLoadTwain/FreeTwain は対で使用してください。
IKScanFreeTwain を実行する場合は、事前に IKScanTerminate を実行するようにしてください。

IKScanGetBitDepth

【機能】
データソースがサポートしている画素ビット数を取得します。

【関数書式】
(1)C++Builder
DWORD IKScanGetBitDepth(HANDLE AppHandle, HANDLE SrcHandle, LPCSTR ScanDsName, WORD PixelType, WORD *List, DWORD ListNum);
(2)Delphi
function IKScanGetBitDepth(AppHandle, SrcHandle: THandle; ScanDsName: PAnsiChar; PixelType: Word; var List: Word; ListNum: DWORD): DWORD;

【引数】

名称	内容	
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル	
SrcHandle	**IKScanOpenDS** で取得したデータソースハンドルもしくは 0	
ScanDsName	データソース名（SrcHandle が 0 の場合は必須）	
PixelType	ピクセルタイプ（0:白黒 2 値、1:グレースケール、2:RGB カラー、3:パレットカラー）	
List	画素ビット数を取得する配列	
	(1)C++Builder	配列の先頭のポインタを渡す
	(2)Delphi	Lt: array [0..2] of DWORD の場合は、Lt[0]を与える
ListNum	List の要素数	

【戻り値】
List と ListNum を設定した場合
取得した数を返します。0 の場合はエラーとなりますのでエラーステータスを確認してください。
List が NULL もしくは ListNum が 0 の場合
取得する際に必要な配列の要素数を返します。データソースによってはピクセルタイプの値に関係なく、サポートしている全ての画素ビット数を取得するものがあります。その場合は、実際の数よりも多い値が返されます。

【解説】
SrcHandle に IKScanOpenDS で取得した値を使用すると、オープン中のデータソースは他のアプリケーションからは利用できなくなり、データソースを占有することができます。
SrcHandle が 0 で ScanDsName に空文字列を指定する場合は、予め IKScanSelect を実行しておいてください。(""や NULL を設定した場合、IKScanSelect で選択されたデータソースとなります。)
PixelType にはデータソースがサポートしている値を設定してください。

IKScanGetCapEnum

【機能】
データソースから指定した項目の設定可能値を取得します。

【関数書式】
(1)C++Builder
DWORD IKScanGetCapEnum(HANDLE AppHandle, HANDLE SrcHandle, LPCSTR ScanDsName, WORD CapNo, WORD *ConType, float *List, LPSTR StringList, DWORD *CurrentIndex, DWORD *DefaultIndex, DWORD ListNum);
(2)Delphi
function IKScanGetCapEnum(AppHandle, SrcHandle: THandle; ScanDsName: PAnsiChar; CapNo: Word; var ConType: Word; var List: Single; StringList: PChar; var CurrentIndex, DefaultIndex: DWORD; ListNum: DWORD): DWORD;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル
SrcHandle	**IKScanOpenDS** で取得したデータソースハンドルもしくは 0
ScanDsName	データソース名（SrcHandle が 0 の場合は必須）
CapNo	取得する項目番号（16 進表記）

0x0100 圧縮方法
0x0101 ピクセルタイプ
0x0102 取り込み単位
0x0103 転送方法
0x1100 明るさの自動化(*1)
0x1101 明るさ
0x1103 コントラスト
0x1108 ガンマ
0x1109 ハーフトーン
0x110a ハイライト
0x110c ファイルフォーマット
0x110e ドロップアウトカラー
0x1110 用紙の向き
0x1113 シャドウ
0x1118 X 方向解像度
0x1119 Y 方向解像度
0x1121 回転角度
0x1122 用紙サイズ
0x1123 しきい値
0x1124 X 方向スケーリング
0x1125 Y 方向スケーリング
0x112c 色深度の減少方法
0x1147 イメージフィルタ
0x1148 ノイズフィルタ
0x1149 オーバースキャン
0x1153 JPEG 品質係数
0x115c 両面合成
0xf001 読み取り速度
0xf002 モアレフィルタ
0xf004 シャープネス
0xf006 拡張 UI モード
0xf007 ダイナミックスレッシュホールド
0xf00b 白紙ページ除去
0xf00e テキストエンハンスメント
※Delphi は 0x を$に置き換えてください。

ConType	項目の型を取得する変数
List	項目値を取得する配列(CapNo がハーフトーン以外を表す場合に使用) (1)C++Builder　配列の先頭のポインタを渡す (2)Delphi　Lt: array [0..2] of Single の場合は、Lt[0]を与える
StringList	文字列を取得する変数(CapNo がハーフトーンを表す場合に使用)
CurrentIndex	現在値を示す List へのインデックス(インデックスは 0 から始まります) List[CurrentIndex] = 現在値
DefaultIndex	デフォルト値を示す List へのインデックス(インデックスは 0 から始まります) List[DefaultIndex] = デフォルト値
ListNum	List の要素数

【戻り値】

CapNo がハーフトーンを表す場合

StringList が NULL(C++Builder),nil(Delphi)のいずれかの場合は、取得する際に必要な文字列のサイズを返します。(終端のヌルは除く)

StringList に変数を指定した場合は、取得した数を返します。

CapNo がハーフトーン以外を表す場合

ConType=3 もしくは 4 の場合は取得した数を返します。

ConType=5 の場合は 1、ConType=6 の場合は 5 を返します。(固定)

List が NULL もしくは ListNum が 0 の場合は、取得する際に必要な配列の要素数を返します。

0 の場合はエラーとなりますので、エラーステータスを確認してください。

【解説】

SrcHandle に IKScanOpenDS で取得した値を使用すると、オープン中のデータソースは他のアプリケーションからは利用できなくなり、データソースを占有することができます。

SrcHandle が 0 で ScanDsName に空文字列を指定する場合は、予め IKScanSelect を実行しておいてください。(""や NULL を設定した場合、IKScanSelect で選択されたデータソースとなります。)

SrcHandle が 0 もしくは本コマンドを実行した後でデータソースをクローズし、再度データソースをオープンして本コマンドを実行した場合、データソースによっては項目に設定された値がクリアされ CurrentIndex と DefaultIndex が同じになる場合があります。

CapNo がハーフトーンを表す場合、List 配列は使用しませんので、ダミー配列や NULL を引数として与えてください。

ListNumも使用しませんので 0 などを与えてください。CapNo がハーフトーン以外を表す場合、StringList は使用しませんので、空文字列や NULL もしくはダミー変数を与えてください。

ConType に設定される値について

(0:関数の戻り値が 0 の場合、3:配列型、4:列挙型、5:単一型、6:範囲型)

CapNo がハーフトーンを表す場合

ConType=3 の場合、StringList に値が設定され、CurrentIndex と DefaultIndex は無効となります。

ConType=4 の場合、StringList に値が設定されます。

ConType=5 の場合、StringList に現在値が設定されます。

ConType=6 の場合はありません。

CapNo がハーフトーン以外を表す場合

ConType=3 の場合、List 配列に値が設定され、CurrentIndex と DefaultIndex は無効となります。

ConType=4 の場合、List 配列に値が設定されます。

ConType=5 の場合、List[0]に現在値が設定されます。

ConType=6 の場合、List[0]に最小値が、List[1]に最大値が、List[2]にステップ値が、List[3]にデフォルト値が、List[4]に現在値がそれぞれ設定されます。

List に設定される値の内容

CapNo が圧縮方法を表す場合:

0:非圧縮、1:PACKBITS、2:GROUP3-1D、3:GROUP3-1DEOL、4:GROUP3-2D、5:GROUP4、6:JPEG
7:LZW、8:JBIG、9:PNG、10:RLE4、11: RLE8、12:BITFIELDS
(2,3,4,5 は FAX[CCITT]、10,11,12 は BMP 形式)

CapNo がピクセルタイプを表す場合:

0:白黒 2 値、1:グレースケール、2:RGB カラー、3:パレットカラー
(データソースによっては 4～8 が設定される場合もあります)

CapNo が取り込み単位を表す場合:
0:インチ、1:cm、2:パイカ、3:ポイント、4:twip、5:ピクセル、6:mm

CapNo が転送方法を表す場合:
0:ネイティブ、1:ファイル、2:メモリ、3:ファイル 2

CapNo がファイルフォーマットを表す場合:
0:TIFF、1:PICT、2:BMP、3:XBM、4:JFIF(JPEG)、5:FPX、6:TIFF MULTI、7:PNG、8:SPIFF、9:EXIF、10:PDF
11:JP2(JPEG2000 Part1)、13:JPX(JPEG2000 Part2)、14:DEJAVU、15:PDF/A(Version 1)、16:PDF/A(Version 2)

CapNo がドロップアウトカラーを表す場合:
0:赤、1:緑、2:青、3:なし、4:白
(データソースによっては 5,6 が設定される場合もあります)

CapNo が用紙の向きを表す場合:
0:0 度(ポートレイト)、1:90 度、2:180 度、3:270 度(ランドスケープ)

CapNo が用紙サイズを表す場合:
0:ユーザ定義サイズ、1:A4、2:JIS B5、3:US レター、4:US リーガル、5:A5、6:ISO B4、7:ISO B6、9:US レジャー
10:US エグゼクティブ、11:A3、12:ISO B3、13:A6、14:C4、15:C5、16:C6、17:4A0、18:2A0、19:A0、20:A1、21:A2
22:A7、23:A8、24:A9、25:A10、26:ISO B0、27:ISO B1、28:ISO B2、29:ISO B5、30:ISO B7、31:ISO B8
32:ISO B9、33:ISO B10、34:JIS B0、35:JIS B1、36:JIS B2、37:JIS B3、38:JIS B4、39:JIS B6、40:JIS B7、41:JIS B8
42:JIS B9、43:JIS B10、44:C0、45:C1、46:C2、47:C3、48:C7、49:C8、50:C9、51:C10、52:US ステイトメント
53:ビジネスカード、54:最大サイズ

CapNo が色深度の減少方法を表す場合:
0:しきい値、1:ハーフトーン、2:カスタムハーフトーン、3:拡散、4:動的しきい値

CapNo がイメージフィルタを表す場合:
0:なし、1:自動、2:LOWPASS、3:BANDPASS、4:HIGHPASS

CapNo がノイズフィルタを表す場合:
0:なし、1:自動、2:LONEPIXEL、3:MAJORITYRULE

CapNo がオーバースキャンを表す場合:
0:指定されたサイズで読み取り、1:指定されたサイズを考慮して自動で読み取り
2:指定されたサイズを基に上下を広げて読み取り、3:指定されたサイズを基に左右を広げて読み取り
4:指定されたサイズを基に上下左右を広げて読み取り

CapNo が両面合成を表す場合:
0:なし(合成しない)、1:指定されたサイズを考慮して自動で読み取り
2:裏面が上で表面が下になる(上下の貼り合わせ)、3:表面が左で裏面が右になる(左右の貼り合わせ)
4:裏面が左で表面が右になる(左右の貼り合わせ)

CapNo が読み取り速度を表す場合:
エプソン製スキャナ用ドライバおよびパナソニック製スキャナ用ドライバご利用時に有効です。
(1)エプソン製スキャナ用ドライバ　0:品質優先、1:速度優先
(2)パナソニック製スキャナ用ドライバ　0:低速、1:通常、2:高速

CapNo がモアレフィルタを表す場合:
エプソン製スキャナ用ドライバ、キヤノン製 DR スキャナ用ドライバ、PFU 製スキャナ用ドライバ、パナソニック製スキャナ用ドライバご利用時に有効です。
(1)エプソン製スキャナ用ドライバ
0:なし、1:標準、2:85lpi(新聞など)、3:133lpi(週刊誌やカタログなど)、4:175lpi(写真など)
(2)キヤノン製 DR スキャナ用ドライバ
1:なし、2:高速、3:高画質
(3)PFU 製スキャナ用ドライバ
0:なし、1:モアレ除去レベル 1、2:モアレ除去レベル 2、3:モアレ除去レベル 3、4:モアレ除去レベル 4
(4)パナソニック製スキャナ用ドライバ
0:なし、1:あり

CapNo がシャープネスを表す場合:
キヤノン製 DR スキャナ用ドライバ、エプソン製スキャナ用ドライバ、パナソニック製スキャナ用ドライバ、PFU 製スキャナ用ドライバご利用時に有効です。
キヤノン製 DR スキャナ用ドライバでは「エッジ強調」、エプソン製スキャナ用ドライバでは「アンシャープマスク」、パナソニック製スキャナ用ドライバでは「画質」、PFU 製スキャナ用ドライバでは「輪郭強調」という表現を用いています。
(1)キヤノン製 DR スキャナ用ドライバ　1 から 5
(2)エプソン製スキャナ用ドライバ　0:なし、1:小、2:中(デフォルト)、3:大
(3)パナソニック製スキャナ用ドライバ　0:なし、1:スムーズ、2:低、3:標準、4:高、5:自動
(4)PFU 製スキャナ用ドライバ　0:なし、1:弱、2:中、3:強

CapNo が拡張 UI モードを表す場合:

エプソン製スキャナ用ドライバご利用時に有効です。

0:前回設定の UI モードで起動、1:プロフェッショナルモードで起動、2:全自動モードで起動、3:ホームモードで起動
4:全自動モードで起動(エラーメッセージを除き、UI 非表示)、5:オフィスモードで起動

CapNo がダイナミックスレッシュホールドを表す場合:

パナソニック製スキャナ用ドライバご利用時に有効です。

0:なし、1:明るい、2:少し明るい、3:標準、4:少し暗い、5:暗い

CapNo がテキストエンハンスメントを表す場合:

キヤノン製 DR スキャナ用ドライバ、エプソン製スキャナ用ドライバご利用時に有効です。

(1)キヤノン製 DR スキャナ用ドライバ
0:適用しない、1:テキストエンハンスメント、2:アドバンスドテキストエンハンスメント、3:高速テキストエンハンスメント
4,5:アドバンスドテキストエンハンスメントⅡ

(2)エプソン製スキャナ用ドライバ
0:適用しない、1:文字くっきり(標準)、2:文字くっきり(強)

CapNo が明るさの自動化、明るさ、コントラスト、ガンマ、ハイライト、シャドウ、X 方向解像度、Y 方向解像度、回転角度、しきい値、X 方向スケーリング、Y 方向スケーリング、JPEG 品質係数、白紙ページ除去を表す場合は設定可能な値となります。
ただし、ConType=6 の場合は IKScanGetCapRange を実行した結果と同じとなります。
(*1)データソースが TWAIN2.0 以降に対応している場合は設定可能な値となりますが、それ以外は現在値となります。

StringList に設定される文字列の内容

CapNo がハーフトーンを表す場合、ハーフトーンを表す名称が設定されます。
名称が 1 つの場合: ″XXX,″
名称が複数の場合: ″XXX,YYY,・・・・・・,″
XXX と YYY は名称を表し、名称の間はカンマで区切られ、終了を表すターミネータもカンマとなります。

【ImageKit6 との違い】
CapNo にファイルフォーマット、ドロップアウトカラー、イメージフィルタ、ノイズフィルタ、JPEG 品質係数、読み取り速度、モアレフィルタ、シャープネス、拡張 UI モード、ダイナミックスレッシュホールドの値が追加されました。

【ImageKit7 との違い】
CapNo に回転角度、オーバースキャン、白紙ページ除去、テキストエンハンスメントの値が追加されました。

【ImageKit8 との違い】
CapNo に両面合成の値が追加されました。

【ImageKit9 との違い】
CapNo に明るさの自動化の値が追加されました。

IKScanGetCapRange

【機能】
データソースから指定した項目の範囲を取得します。

【関数書式】
(1)C++Builder
BOOL IKScanGetCapRange(HANDLE AppHandle, HANDLE SrcHandle, LPCSTR ScanDsName, WORD CapNo, WORD *ConType, PTR_IKSCAN_RANGE Value);
(2)Delphi
function IKScanGetCapRange(AppHandle, SrcHandle: THandle; ScanDsName: PAnsiChar; CapNo: Word; var ConType: Word; var Value: IKSCAN_RANGE): LongBool;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル
SrcHandle	**IKScanOpenDS** で取得したデータソースハンドルもしくは 0
ScanDsName	データソース名（SrcHandle が 0 の場合は必須）
CapNo	取得する項目番号（16 進表記） 0x1101　明るさ 0x1103　コントラスト 0x1108　ガンマ 0x110a　ハイライト 0x1113　シャドウ 0x1118　X 方向解像度 0x1119　Y 方向解像度 0x1121　回転角度 0x1123　しきい値 0x1124　X 方向スケーリング 0x1125　Y 方向スケーリング 0x1153　JPEG 品質係数 0xf004　シャープネス 0xf00b　白紙ページ除去 0xf00d　焦点位置 ※Delphi は 0x を$に置き換えてください。
ConType	項目の型を取得する変数
Value	範囲を取得する構造体（ユーザ定義型）変数

【戻り値】
成功の場合は True(0 以外)、失敗の場合は False(0)を返します。

【解説】
SrcHandle に IKScanOpenDS で取得した値を使用すると、オープン中のデータソースは他のアプリケーションからは利用できなくなり、データソースを占有することができます。
SrcHandle が 0 で ScanDsName に空文字列を指定する場合は、予め IKScanSelect を実行しておいてください。(""や NULL を設定した場合、IKScanSelect で選択されたデータソースとなります。)
Value は、範囲を取得する構造体（ユーザ定義型）変数です。
IKSCAN_RANGE については、**「Ik10Scan.dll/Ik10ScanA.dll/Ik10Scan64.dll/Ik10Scan64A.dll」の構造体の定義**をご覧ください。

ConType に設定される値について
（0：関数の戻り値が False(0)の場合、3:配列型、4:列挙型、5：単一型、6：範囲型）
ConType=3 または 4 の場合、Value.Max に項目を取得する際に必要な要素数が設定されますので、その要素数を基に IKScanGetCapEnum を実行してください。
ConType=5 の場合、Value.Current に現在値が、Value.Default にデフォルト値が設定されます。

ConType=6 の場合、Value の全てのメンバーに値が設定されます。

【ImageKit6 との違い】
CapNo に JPEG 品質係数の値が追加されました。

【ImageKit7 との違い】
CapNo に回転角度、シャープネス、白紙ページ除去、焦点位置の値が追加されました。

IKScanGetDSInfo

【機能】
指定したデータソースの情報を取得します。

【関数書式】
(1)C++Builder
BOOL IKScanGetDSInfo(HANDLE AppHandle, LPCSTR ScanDsName, PTR_IKSCAN_DATASOURCEINFO DS);
(2)Delphi
function IKScanGetDSInfo(AppHandle: THandle; ScanDsName: PAnsiChar; var DS: IKSCAN_DATASOURCEINFO): LongBool;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル
ScanDsName	データソース名
DS	データソースの情報を取得する構造体(ユーザ定義型)変数

【戻り値】
成功の場合は True(0 以外)、失敗の場合は False(0)を返します。

【解説】
ScanDSName に""や NULL を設定した場合は、IKScanSelect で選択されたデータソースが対象となります。DS は、データソースの情報を取得する構造体(ユーザ定義型)変数です。
IKSCAN_DATASOURCEINFO については、**「Ik10Scan.dll/Ik10ScanA.dll/Ik10Scan64.dll/Ik10Scan64A.dll」の構造体の定義**をご覧ください。

【ImageKit6 との違い】
IKSCAN_DATASOURCEINFO 構造体の一部のメンバー変数の型が 4 バイト型から 2 バイト型に変更されました。

IKScanGetMinimumSize,IKScanGetMinimumSizeEx

【機能】
データソースから取り込める最小サイズを取得します。

【関数書式】
(1)C++Builder
BOOL **IKScanGetMinimumSize**(HANDLE AppHandle, HANDLE SrcHandle, LPCSTR ScanDsName, WORD ScanMode, WORD UnitMode, float *Width, float *Height);
BOOL **IKScanGetMinimumSizeEx**(HANDLE AppHandle, HANDLE SrcHandle, LPCSTR ScanDsName, WORD ScanMode, WORD UnitMode, int Xdpi, int Ydpi, float *Width, float *Height);
(2)Delphi
function **IKScanGetMinimumSize**(AppHandle, SrcHandle: THandle; ScanDsName: PAnsiChar; ScanMode, UnitMode: Word; var Width, Height: Single): LongBool;
function **IKScanGetMinimumSizeEx**(AppHandle, SrcHandle: THandle; ScanDsName: PAnsiChar; ScanMode, UnitMode: Word; Xdpi, Ydpi: Integer; var Width, Height: Single): LongBool;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル
SrcHandle	**IKScanOpenDS** で取得したデータソースハンドルもしくは 0
ScanDsName	データソース名（SrcHandle が 0 の場合は必須）
ScanMode	取り込みモード（0:反射原稿-原稿台、1:反射原稿-ADF 片面、2:反射原稿-ADF 両面）
UnitMode	取り込み単位（0:インチ、1:cm、2:パイカ、3:ポイント、4:twip、5:ピクセル、6:mm）
Xdpi	UnitMode が 5 の場合の X 方向解像度(DPI 単位)、**IKScanGetMinimumSizeEx** で使用
Ydpi	UnitMode が 5 の場合の Y 方向解像度(DPI 単位)、**IKScanGetMinimumSizeEx** で使用
Width	取り込める最小幅を取得する変数
Height	取り込める最小高さを取得する変数

【戻り値】
成功の場合は True(0 以外)、失敗の場合は False(0)を返します。

【解説】
SrcHandle に IKScanOpenDS で取得した値を使用すると、オープン中のデータソースは他のアプリケーションからは利用できなくなり、データソースを占有することができます。
ScanDsName に空文字列を指定する場合は、予め IKScanSelect を実行しておいてください。(""や NULL を設定した場合、IKScanSelect で選択されたデータソースとなります。)
UnitMode にはデータソースがサポートしている値を設定してください。
Width、Height には ScanMode および UnitMode に応じた値が設定されます。**IKScanGetMinimumSize** では UnitMode が 5（ピクセル）の場合は現在の解像度（DPI）に応じた値となりますが、**IKScanGetMinimumSizeEx** では Xdpi と Ydpi に応じた値となります。

【ImageKit7 との違い】
IKScanGetMinimumSizeEx が追加されました。

IKScanGetPhysicalSize,IKScanGetPhysicalSizeEx

【機能】
データソースから取り込める最大物理サイズを取得します。

【関数書式】
(1)C++Builder
BOOL **IKScanGetPhysicalSize**(HANDLE AppHandle, HANDLE SrcHandle, LPCSTR ScanDsName, WORD ScanMode, WORD UnitMode, float *Width, float *Height);
BOOL **IKScanGetPhysicalSizeEx**(HANDLE AppHandle, HANDLE SrcHandle, LPCSTR ScanDsName, WORD ScanMode, WORD UnitMode, int Xdpi, int Ydpi, float *Width, float *Height);
(2)Delphi
function **IKScanGetPhysicalSize**(AppHandle, SrcHandle: THandle; ScanDsName: PAnsiChar; ScanMode, UnitMode: Word; var Width, Height: Single): LongBool;
function **IKScanGetPhysicalSizeEx**(AppHandle, SrcHandle: THandle; ScanDsName: PAnsiChar; ScanMode, UnitMode: Word; Xdpi, Ydpi: Integer; var Width, Height: Single): LongBool;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル
SrcHandle	**IKScanOpenDS** で取得したデータソースハンドルもしくは 0
ScanDsName	データソース名（SrcHandle が 0 の場合は必須）
ScanMode	取り込みモード（0:反射原稿-原稿台、1:反射原稿-ADF 片面、2:反射原稿-ADF 両面 7:透過原稿-ポジフィルム、フィルムホルダ使用、8:透過原稿--ポジフィルム、フィルムエリアガイド使用）
UnitMode	取り込み単位（0:インチ、1:cm、2:パイカ、3:ポイント、4:twip、5:ピクセル、6:mm）
Xdpi	UnitMode が 5 の場合の X 方向解像度(DPI 単位)、**IKScanGetPhysicalSizeEx** で使用
Ydpi	UnitMode が 5 の場合の Y 方向解像度(DPI 単位)、**IKScanGetPhysicalSizeEx** で使用
Width	取り込める最大物理幅を取得する変数
Height	取り込める最大物理高さを取得する変数

【戻り値】
成功の場合は True(0 以外)、失敗の場合は False(0)を返します。

【解説】
SrcHandle に IKScanOpenDS で取得した値を使用すると、オープン中のデータソースは他のアプリケーションからは利用できなくなり、データソースを占有することができます。
ScanDsName に空文字列を指定する場合は、予め IKScanSelect を実行しておいてください。(""や NULL を設定した場合、IKScanSelect で選択されたデータソースとなります。)
UnitMode にはデータソースがサポートしている値を設定してください。
Width、Height には ScanMode および UnitMode に応じた値が設定されます。**IKScanGetPhysicalSize** では UnitMode が 5（ピクセル）の場合は現在の解像度（DPI）に応じた値となりますが、**IKScanGetPhysicalSizeEx** では Xdpi と Ydpi に応じた値となります。

【ImageKit5 との違い】

関数名	**引数の並び**
IK5ScanGetPhysicalSize:	hWnd, ScanDsName, ScanMode, Width, Height
IKScanGetPhysicalSize:	AppHandle, SrcHandle, ScanDsName, ScanMode, UnitMode, Width, Height

IK6 から hWnd を IKScanInitialize で渡し、その戻り値の AppHandle を使用します。

【ImageKit7 との違い】
IKScanGetPhysicalSizeEx が追加されました。

IKScanGetPixelType

【機能】
データソースがサポートしているピクセルタイプを取得します。

【関数書式】
(1)C++Builder
DWORD IKScanGetPixelType(HANDLE AppHandle, HANDLE SrcHandle, LPCSTR ScanDsName, WORD *List, WORD ListNum);
(2)Delphi
function IKScanGetPixelType(AppHandle, SrcHandle: THandle; ScanDsName: PAnsiChar; var List: Word; ListNum: Word): DWORD;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル
SrcHandle	**IKScanOpenDS** で取得したデータソースハンドルもしくは 0
ScanDsName	データソース名（SrcHandle が 0 の場合は必須）
List	ピクセルタイプを取得する配列 (1)C++Builder　配列の先頭のポインタを渡す (2)Delphi　List: array [0..3] of Word の場合は、引数に List[0]を与える
ListNum	List の要素数

【戻り値】
List と ListNum を設定した場合
取得した数を返します。0 の場合はエラーとなりますのでエラーステータスを確認してください。
List が NULL もしくは ListNum が 0 の場合
取得する際に必要な配列の要素数を返します。

【解説】
SrcHandle に IKScanOpenDS で取得した値を使用すると、オープン中のデータソースは他のアプリケーションからは利用できなくなり、データソースを占有することができます。
List 配列に取得する値は 0～3 となり、IKSCAN_EXEC のメンバーの PixelType に設定する値と同様となります。
IKSCAN_EXEC については、**「Ik10Scan.dll/Ik10ScanA.dll/Ik10Scan64.dll/Ik10Scan64A.dll」の構造体の定義**をご覧ください。
ScanDsName に空文字列を指定する場合は、予め IKScanSelect を実行しておいてください。(""や NULL を設定した場合、IKScanSelect で選択されたデータソースとなります。)
このコマンドは ImageKit5 の互換性のために用意してありますが、同等な機能を提供するものとして IKScanGetCapEnum もあります。

【ImageKit5 との違い】

関数名	**引数の並び**
IK5ScanGetPixeltype:	hWnd, ScanDsName, List, ListNum
IKScanGetPixelType:	AppHandle, SrcHandle, ScanDsName, List, ListNum

関数名の Pixeltype の t が大文字に変更されました。
IK6 から hWnd を IKScanInitialize で渡し、その戻り値の AppHandle を使用します。
戻り値についても成功もしくは失敗から、要素数を返すように変更されています。

IKScanInitialize

【機能】
TWAIN の初期化を行います。

【関数書式】
(1)C++Builder
HANDLE IKScanInitialize(HWND hWnd, WORD MajorNum, WORD MinorNum, LPCSTR VersionInfo, LPCSTR Manufacturer, LPCSTR ProductFamily, LPCSTR ProductName);
(2)Delphi
function IKScanInitialize(hWnd: HWND; MajorNum, MinorNum: Word; VersionInfo, Manufacturer, ProductFamily, ProductName: PAnsiChar): THandle;

【引数】

名称	内容
hWnd	フォームのウィンドウハンドル
MajorNum	アプリケーションのメジャーバージョン番号(バージョンが 1.00 の場合は 1)
MinorNum	アプリケーションのマイナーバージョン番号(バージョンが 1.00 の場合は 0)
VersionInfo	アプリケーションのバージョン情報(32 バイト以内)
Manufacturer	製造者名(32 バイト以内)
ProductFamily	製品ファミリー名(32 バイト以内)
ProductName	製品名(32 バイト以内)

【戻り値】
アプリケーションハンドルを返します。(実行に失敗した場合は 0)

【解説】
TWAIN による機能を使用する場合、必ず実行する必要があります。
MajorNum、MinorNum、VersionInfo、Manufacturer、ProductFamily、ProductName は TWAIN に対してアプリケーションの情報を通知するために使用されます。
戻り値のアプリケーションハンドルはスキャン関連の関数で使用します。

IKScanIsCapSupported

【機能】
データソースから指定した項目がサポートされているかどうかを確認します。

【関数書式】
(1)C++Builder
BOOL IKScanIsCapSupported(HANDLE AppHandle, HANDLE SrcHandle, LPCSTR ScanDsName, WORD CapNo, int *Value);
(2)Delphi
function IKScanIsCapSupported(AppHandle, SrcHandle: THandle; ScanDsName: PAnsiChar; CapNo: Word; var Value: Integer): LongBool;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル
SrcHandle	**IKScanOpenDS** で取得したデータソースハンドルもしくは 0
ScanDsName	データソース名（SrcHandle が 0 の場合は必須）
CapNo	取得する項目番号（16 進表記） 0x1002　ドキュメントフィーダー機能 0x1003　ドキュメントフィーダーに用紙があるかどうかを確認する機能 0x100b　インジケータ 0x100e　UI コントロール機能 0x100f　デバイスの稼動状態の確認機能 0x1011　サムネイル画像の転送機能 0x1012　ドキュメントフィーダーの両面機能 0x1014　設定用 UI 表示機能 0x1015　UI 設定値の取得/設定機能 0x1100　明るさの自動化 0x112d　未定義サイズ 0x112e　デジタルカメラ内の画像枚数 0x1150　自動領域切り出し機能 0x1151　デスキュー（傾き補正） 0xf003　モアレフィルタ使用時の最大有効解像度 0xf005　ADF を使用した両面スキャン時の裏面回転 0xf008　白黒/カラー自動判別読み取り機能 0xf009　マルチストリーム 0xf00a　文字向き検知または自動回転 0xf00c　パンチ穴除去 0xf00f　境界補正 0xf010　ガンマ補正 ※Delphi は 0x を$に置き換えてください。
Value	状態を取得する変数

【戻り値】
成功の場合は True(0 以外)、失敗の場合は False(0)を返します。

【解説】
SrcHandle に IKScanOpenDS で取得した値を使用すると、オープン中のデータソースは他のアプリケーションからは利用できなくなり、データソースを占有することができます。
SrcHandle が 0 で ScanDsName に空文字列を指定する場合は、予め IKScanSelect を実行しておいてください。（""や NULL を設定した場合、IKScanSelect で選択されたデータソースとなります。）

Value に設定される値について
CapNo がドキュメントフィーダー機能を示す場合、0:フィーダー使用不可（原稿台使用可）, 1:フィーダー使用可, 2: 原稿台

とフィーダー使用可。
CapNo がドキュメントフィーダーに用紙があるかどうかの機能を示す場合、0:用紙なし, 1:用紙あり。
CapNo がインジケータを示す場合、0:インジケータ使用不可, 1:インジケータ使用可。
CapNo が UI コントロール機能を示す場合、0:メーカ提供 UI のみ, 1:メーカ提供 UI 非表示可。
CapNo がデバイスの稼動状態の確認機能を示す場合、現在の状態を表します。(0:応答なし, 1:使用可)
CapNo がサムネイル画像の転送機能を示す場合、0:サムネイル転送不可, 1:サムネイル転送可。
CapNo がドキュメントフィーダーの両面機能を示す場合、0:両面スキャン不可, 1:1 回の読み取りで両面スキャン、2:2 回の読み取りで両面スキャン。
CapNo が設定用 UI 表示機能を示す場合、0:設定用 UI 表示不可, 1:設定用 UI 表示可。
CapNo が UI 設定値の取得/設定機能を示す場合、0:UI 設定値の取得/設定不可, 1: UI 設定値の取得/設定可。
CapNo が明るさの自動化を示す場合、0:明るさの自動不可, 1:明るさの自動可。
CapNo が未定義サイズを示す場合、一般的には「0:未定義サイズ使用不可, 1:未定義サイズ使用可」となりますが、**エプソン製スキャナ用ドライバご利用時**には次の値が設定されます。「0:使用不可, 1:原稿台のみ可, 2:ADF のみ可, 3:原稿台,ADF とも可」。
CapNo がデジタルカメラ内の画像枚数を示す場合、撮影した画像の数を表します。
CapNo が自動領域切り出し機能を示す場合、0:領域切り出し不可, 1:領域切り出し可。
CapNo がデスキュー(傾き補正)を示す場合、0:デスキュー使用不可, 1:デスキュー使用可, 2:デスキュー滑らかオプション使用可。**2 が設定されるのはパナソニック製スキャナ用ドライバご利用時のみ**ですが、その場合は 1 も使用可能です。
CapNo がモアレフィルタ使用時の最大有効解像度を示す場合、最大有効解像度を表します。**エプソン製スキャナ用ドライバご利用時に有効です**。
CapNo が ADF を使用した両面スキャン時の裏面回転を示す場合、0:裏面回転不可, 1:裏面回転可。**キヤノン製 DR スキャナ用ドライバ、エプソン製スキャナ用ドライバ『EPSON Scan』ご利用時に有効です**。
CapNo が白黒/カラー自動判別読み取り機能を示す場合、0:白黒/カラー自動判別読み取り不可, 1:白黒/カラー自動判別読み取り可。**キヤノン製 DR スキャナ用ドライバ、エプソン製スキャナ用ドライバ、パナソニック製スキャナ用ドライバ、PFU 製スキャナ用ドライバご利用時に有効です**。
CapNo がマルチストリームを示す場合、0:マルチストリーム不可, 1:マルチストリーム可。マルチストリームとは 1 枚の原稿から 2 つの異なるピクセルタイプの画像を取得することです。**キヤノン製 DR スキャナ用ドライバ、パナソニック製スキャナ用ドライバ、PFU 製スキャナ用ドライバ、エプソン製スキャナ用ドライバご利用時に有効です**。
CapNo が文字向き検知または自動回転を示す場合、0:文字向き検知または自動回転不可, 1:文字向き検知または自動回転可。文字向き検知については**キヤノン製 DR スキャナ用ドライバ、パナソニック製スキャナ用ドライバご利用時に有効です**。
キヤノン製 DR スキャナ用ドライバでは「文字向き検知」、パナソニック製スキャナ用ドライバでは「原稿方向補正」という表現を用いています。
また、TWAIN 仕様書に記載されている自動回転に対応しているドライバについても確認できます。
CapNo がパンチ穴除去を示す場合、0:パンチ穴除去不可, 1:パンチ穴除去可。**キヤノン製 DR スキャナ用ドライバ、エプソン製スキャナ用ドライバ、パナソニック製スキャナ用ドライバ、PFU 製スキャナ用ドライバご利用時に有効です**。
CapNo が境界補正を示す場合、0:境界補正不可, 1:境界補正可。**エプソン製スキャナ用ドライバ、パナソニック製スキャナ用ドライバご利用時に有効です**。
CapNo がガンマ補正を示す場合、0:ガンマ補正不可, 1:ガンマ補正可。**エプソン製スキャナ用ドライバ**ご利用時に有効です。

※関数の戻り値が False(0)の場合には Value は設定されません。

【ImageKit6 との違い】
CapNo に未定義サイズ、自動領域切り出し機能、デスキュー、モアレフィルタ使用時の最大有効解像度、ADF を使用した両面スキャン時の裏面回転の値が追加されました。

【ImageKit7 との違い】
CapNo に設定用 UI 表示機能、UI 設定値の取得/設定機能、マルチストリーム、文字向き検知または自動回転、パンチ穴除去の値が追加されました。また、ドキュメントフィーダー機能で Value に新たに 2 が設定されるようになりました。

【ImageKit8 との違い】
CapNo に境界補正、ガンマ補正の値が追加されました。

【ImageKit9 との違い】
CapNo に明るさの自動化の値が追加されました。

IKScanList

【機能】
データソースを列挙します。

【関数書式】
(1)C++Builder
BOOL IKScanList(HANDLE AppHandle, LPSTR List, short *SourceCount);
(2)Delphi
function IKScanList(AppHandle: THandle; List: PAnsiChar; var SourceCount: Smallint): LongBool;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル
List	データソースの列挙文字列を取得する変数
SourceCount	インストールされているデータソースの数を取得する変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
※List には大きめのサイズを確保してください。
(1)C++Builder
char List[1024];
(2)Delphi
List: array[0..1023] of AnsiChar;

戻り値が True(0 以外)の場合、List 文字列は"カレント,xxxxx,xxxxx,xxxxx,・・・・・・・・,"となり、名称はカンマで区切られて設定されます(終了は","です)。データソースが何も選択されていない場合は","から始まります。
カレントは現在選択されているデータソースとなります。

SourceCount は List 文字列の最初の項目(カレント)を除いた項目数と同じです。

【ImageKit5 との違い】

関数名	**引数の並び**
IK5ScanList:	hWnd, List
IKScanList:	AppHandle, List, SourceCount

IK6 から hWnd を IKScanInitialize で渡し、その戻り値の AppHandle を使用します。

IKScanListLen

【機能】
データソースを列挙した文字列の長さを返します。

【関数書式】
(1)C++Builder
int IKScanListLen(HANDLE AppHandle);
(2)Delphi
function IKScanListLen(AppHandle: THandle): Integer;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル

【戻り値】
データソースを列挙した文字列の長さ(0 は失敗)

【解説】
データソースを列挙した文字列の長さを返します。(終端のヌルは除く)

【ImageKit5 との違い】

関数名	**引数の並び**
IK5ScanListLen:	hWnd
IKScanListLen:	AppHandle

IK6 から hWnd を IKScanInitialize で渡し、その戻り値の AppHandle を使用します。

IKScanLoadTwain

【機能】
TWAIN DLL をロードします。

【関数書式】
(1)C++Builder
BOOL IKScanLoadTwain(LPCSTR FileName);
(2)Delphi
function IKScanLoadTwain(FileName: PAnsiChar): LongBool;

【引数】

名称	内容
FileName	TWAIN DLL のファイル名 (フルパス)

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
32 ビット用の DLL で twain_32.dll(Windows フォルダ)を使用する場合や 64 ビット用の DLL で 64 ビット用の TWAINDSM.dll(Windows¥System32 フォルダ)を使用する場合は当関数を実行する必要はありません (ImageKit 内で自動的にロード/解放が行われます)。
32 ビット用の DLL で 32 ビット用の TWAINDSM.dll を使用する場合に当関数を実行してください。
IKScanLoadTwain は IKScanInitialize より前に実行してください。

IKScanOpenDS

【機能】
データソースをオープンします。

【関数書式】
(1)C++Builder
HANDLE IKScanOpenDS(HANDLE AppHandle, LPCSTR ScanDsName);
(2)Delphi
function IKScanOpenDS(AppHandle: THandle; ScanDsName: PAnsiChar): THandle;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル
ScanDsName	オープンするデータソース名

【戻り値】
データソースハンドルを返します。(実行に失敗した場合は 0)

【解説】
ScanDsName に空文字列を指定する場合は、予め IKScanSelect を実行しておいてください。
(""や NULL を設定した場合、IKScanSelect で選択されたデータソースとなります。)
IKScanOpenDS を実行すると、IKScanCloseDS を実行する前までの間、オープンしたデータソースを占有することができ、他のアプリケーションからは利用できなくなります。

IKScanSelect

【機能】
データソースの選択を実行します。

【関数書式】
(1)C++Builder
BOOL IKScanSelect(HANDLE AppHandle, PTR_IKSCAN_DATASOURCEINFO SelectDS);
(2)Delphi
function IKScanSelect(AppHandle: THandle; var SelectDS: IKSCAN_DATASOURCEINFO): LongBool;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル
SelectDS	選択されたデータソースの情報を取得する変数

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
IKScanSelect を実行すると、TWAIN で用意されているデータソースの選択用ダイアログが表示され、データソースの選択が可能になります。成功した場合、SelectDS に選択されたデータソースの情報が設定されます。
IKSCAN_DATASOURCEINFO については、**「Ik10Scan.dll/Ik10ScanA.dll/Ik10Scan64.dll/Ik10Scan64A.dll」の構造体の定義**をご覧ください。

【ImageKit5 との違い】

関数名	**引数の並び**
IK5ScanSelect:	hWnd
IKScanSelect:	AppHandle, SelectDS

IK6 から hWnd を IKScanInitialize で渡し、その戻り値の AppHandle を使用します。

【ImageKit6 との違い】
IKSCAN_DATASOURCEINFO 構造体の一部のメンバー変数の型が 4 バイト型から 2 バイト型に変更されました。

IKScanTerminate

【機能】
TWAIN の終了処理を行います。

【関数書式】
(1)C++Builder
BOOL IKScanTerminate(HANDLE AppHandle);
(2)Delphi
function IKScanTerminate(AppHandle: THandle): LongBool;

【引数】

名称	内容
AppHandle	**IKScanInitialize** で取得したアプリケーションハンドル

【戻り値】
成功の場合は True(0 以外)、失敗の場合は False(0)を返します。

【解説】
TWAIN の終了処理を行うと共に、IKScanInitialize で取得したアプリケーションハンドルを解放します。AppHandle に無効な値を設定した場合はエラーとなります。

１－７. Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll

ベクトルイメージを扱う上で基本となる機能を提供します。

●DLL 関数コマンド一覧（アルファベット順）

関数名	内容
IKCreateVectImage	EMF イメージを新規に作成
IKCreateVectImageEx	ベクトルイメージを新規に作成
IKDrawVectObject	ベクトルイメージの描画
IKVectorGdipEnd	GDI+の終了処理
IKVectorGdipStart	GDI+の開始処理
IKVectorToRaster	ベクトルイメージをラスタイメージに変換

IKCreateVectImage,IKCreateVectImageEx

【機能】
新規にベクトルイメージを作成します。

【関数書式】
(1)C++Builder
HANDLE IKCreateVectImage(long Width, long Height, long Xdpi, long Ydpi);
HANDLE IKCreateVectImageEx(long Type, long Width, long Height, long Xdpi, long Ydpi);
(2)Delphi
function IKCreateVectImage(Width, Height, Xdpi, Ydpi: Longint): THandle;
function IKCreateVectImageEx(Type, Width, Height, Xdpi, Ydpi: Longint): THandle;

【引数】

名称	内容
Type	作成するベクトルイメージのタイプ(0:WMF,1:EMF,2:DXF,3:SVG,4:SXF) ※IKCreateVectImageEx の場合に使用
Width	作成するイメージの幅(ピクセル)
Height	作成するイメージの高さ(ピクセル)
Xdpi	作成するイメージの横方向の 1 インチあたりのピクセル数
Ydpi	作成するイメージの縦方向の 1 インチあたりのピクセル数

【戻り値】
作成したベクトルイメージのメモリハンドル(実行に失敗した場合は、0 が返されます)

【解説】
ベクトルイメージをメモリ上に作成します。
IKCreateVectImage は EMF イメージの作成ですが、**IKCreateVectImageEx** では指定したベクトルイメージを作成できます。

IKDrawVectObject

【機能】
ベクトルイメージをデバイスコンテキストに描画します。

【関数書式】
(1)C++Builder
BOOL IKDrawVectObject(HDC hDC, HANDLE hMbh, int DstOrgX, int DstOrgY, int SrcOrgX, int SrcOrgY, double XScale, double YScale, BOOL Black);
(2)Delphi
function IKDrawVectObject(hDC: HDC; hMbh: THandle; DstOrgX, DstOrgY, SrcOrgX, SrcOrgY: Integer; XScale, YScale: Double; Black: LonBool): LongBool;

【引数】

名称	内容
hDC	描画先のデバイスコンテキスト
hMbh	ベクトルイメージのメモリハンドル
DstOrgX	デバイスコンテキストに対する描画開始 X 座標
DstOrgY	デバイスコンテキストに対する描画開始 Y 座標
SrcOrgX	描画対象となるベクトルイメージの開始 X 座標
SrcOrgY	描画対象となるベクトルイメージの開始 Y 座標
XScale	X 方向のスケール（実寸は 1.0）
YScale	Y 方向のスケール（実寸は 1.0）
Black	hMbh が DXF を表す場合、白を黒に置き換えて描画［ True(0 以外):する　False(0):しない ］

【戻り値】
実行に失敗した場合は False(0)、成功した場合は True(0 以外)が返されます。

【解説】
DstOrgX,DstOrgY,SrcOrgX,SrcOrgY はピクセル単位で与えます。
hMbh を全て描画する場合は SrcOrgX,SrcOrgY に 0 を設定してください。
hMbh が DXF を表す場合、Black が有効になります。
当関数を実行する場合は、事前に IKVectorGdipStart 関数を実行する必要があります。

IKVectorGdipEnd

【機能】
GDI+の終了処理を行います。

【関数書式】
(1)C++Builder
void IKVectorGdipEnd(ULONG_PTR token);
(2)Delphi
procedure IKVectorGdipEnd(token: DWORD);

【引数】

名称	内容
token	**IKVectorGdipStart** で取得したトークン

【戻り値】
ありません。

【解説】
IKDrawVectObject,IKVectorToRaster 関数および **Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll** のベクトルイメージの入出力関数を使用する場合に使用します。

IKVectorGdipStart

【機能】
GDI+の初期化処理を行います。

【関数書式】
(1)C++Builder
ULONG_PTR IKVectorGdipStart(void);
(2)Delphi
function IKVectorGdipStart(): DWORD;

【引数】
ありません。

【戻り値】
トークン（失敗した場合は、0 が返されます）。

【解説】
IKDrawVectObject,IKVectorToRaster 関数および **Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll** のベクトルイメージの入出力関数を使用する場合に使用します。

IKVectorToRaster

【機能】
ベクトルイメージをラスタイメージに変換します。

【関数書式】
(1)C++Builder
HANDLE IKVectorToRaster(HANDLE hMbh, short BitCount, BOOL Black, BYTE BackRed, BYTE BackGre, BYTE BackBlu);
(2)Delphi
function IKVectorToRaster(hMbh: THandle; BitCount: Smallint; Black: LongBool; BackRed, BackGre, BackBlu: Byte): THandle;

【引数】

名称	内容
hMbh	ベクトルイメージのメモリハンドル
BitCount	ラスタイメージに変換する際のビット数(1,4,8,16,24,32)
Black	hMbh が DXF を表す場合、白を黒に置き換えて変換 [True(0 以外):する False(0):しない]
BackRed	背景色(赤)
BackGre	背景色(緑)
BackBlu	背景色(青)

【戻り値】
ラスタイメージのメモリハンドル(実行に失敗した場合は 0 が返されます。)

【解説】
hMbh が DXF を表す場合、Black の設定が有効になります。
当関数を実行する場合は、事前に IKVectorGdipStart 関数を実行する必要があります。

【ImageKit6 との違い】
関数の名称が IKVectToDib から変更されました。

2. 作成したアプリケーションの配布

ImageKit10 を使用して作成したアプリケーションをユーザに配布する場合は、そのアプリケーションの実行時に次のファイルが必要になりますので、忘れずに配布リストに加えてください。

32 ビットの Unicode 版
- Ik10Com.dll（必ず必要）
- Ik10Effect.dll（エフェクト処理を行う場合に必要）
- Ik10File.dll（ファイル関連の DLL を扱う場合に必ず必要）
  - Ik10Bmp.dll（BMP 形式のロード、保存を行う場合に必要）
  - Ik10Dxf.dll（DXF 形式のロード、保存を行う場合に必要）
  - Ik10Emf.dll（EMF 形式のロード、保存を行う場合に必要）
  - Ik10Fpx.dll（FPX 形式のロード、保存を行う場合に必要）
  - Ik10Gif.dll（GIF 形式のロード、保存を行う場合に必要）
  - Ik10J2k.dll（JPEG2000 形式のロード、保存を行う場合に必要）
  - Ik10Jpeg.dll（JPEG 形式のロード、保存を行う場合に必要）
  - Ik10Pcx.dll（PCX 形式のロード、保存を行う場合に必要）
  - Ik10Pdf.dll（PDF 形式の保存を行う場合に必要）
  - Ik10Png.dll（PNG 形式のロード、保存を行う場合に必要）
  - Ik10Svg.dll（SVG 形式のロード、保存を行う場合に必要）
  - Ik10SxfP21.dll（SXF p21 形式のロード、保存を行う場合に必要）
  - Ik10SxfSfc.dll（SXF sfc 形式のロード、保存を行う場合に必要）
  - Ik10Tiff.dll（TIFF 形式のロード、保存を行う場合に必要）
  - Ik10TransFile.dll（FTP・HTTP 転送を行う場合に必要）
  - Ik10Wmf.dll（WMF 形式のロード、保存を行う場合に必要）
- Ik10Print.dll（スクリーンやプリンタおよびメモリハンドルに描画する場合に必要）
- Ik10RasToVect.dll（ラスタイベージからベクトルイメージへの変換を行う場合に必要）
- Ik10Scan.dll（TWAIN を使用したイメージの取り込みの場合に必要）
- Ik10VectCom.dll（ベクトルイメージを扱う場合に必要）

32 ビットの Ansi 版
- Ik10ComA.dll（必ず必要）
- Ik10EffectA.dll（エフェクト処理を行う場合に必要）
- Ik10FileA.dll（ファイル関連の DLL を扱う場合に必ず必要）
  - Ik10BmpA.dll（BMP 形式のロード、保存を行う場合に必要）
  - Ik10DxfA.dll（DXF 形式のロード、保存を行う場合に必要）
  - Ik10EmfA.dll（EMF 形式のロード、保存を行う場合に必要）
  - Ik10FpxA.dll（FPX 形式のロード、保存を行う場合に必要）
  - Ik10GifA.dll（GIF 形式のロード、保存を行う場合に必要）
  - Ik10J2kA.dll（JPEG2000 形式のロード、保存を行う場合に必要）
  - Ik10JpegA.dll（JPEG 形式のロード、保存を行う場合に必要）
  - Ik10PcxA.dll（PCX 形式のロード、保存を行う場合に必要）
  - Ik10PdfA.dll（PDF 形式の保存を行う場合に必要）
  - Ik10PngA.dll（PNG 形式のロード、保存を行う場合に必要）
  - Ik10SvgA.dll（SVG 形式のロード、保存を行う場合に必要）
  - Ik10SxfP21A.dll（SXF p21 形式のロード、保存を行う場合に必要）
  - Ik10SxfSfcA.dll（SXF sfc 形式のロード、保存を行う場合に必要）
  - Ik10TiffA.dll（TIFF 形式のロード、保存を行う場合に必要）
  - Ik10TransFileA.dll（FTP・HTTP 転送を行う場合に必要）
  - Ik10WmfA.dll（WMF 形式のロード、保存を行う場合に必要）
- Ik10PrintA.dll（スクリーンやプリンタおよびメモリハンドルに描画する場合に必要）
- Ik10RasToVectA.dll（ラスタイベージからベクトルイメージへの変換を行う場合に必要）
- Ik10ScanA.dll（TWAIN を使用したイメージの取り込みの場合に必要）
- Ik10VectComA.dll（ベクトルイメージを扱う場合に必要）

64 ビットの Unicode 版
- Ik10Com64.dll（必ず必要）

Ik10Effect64.dll（エフェクト処理を行う場合に必要）
Ik10File64.dll（ファイル関連の DLL を扱う場合に必ず必要）
Ik10Bmp64.dll（BMP 形式のロード、保存を行う場合に必要）
Ik10Dxf64.dll（DXF 形式のロード、保存を行う場合に必要）
Ik10Emf64.dll（EMF 形式のロード、保存を行う場合に必要）
Ik10Fpx64.dll（FPX 形式のロード、保存を行う場合に必要）
Ik10Gif64.dll（GIF 形式のロード、保存を行う場合に必要）
Ik10J2k64.dll（JPEG2000 形式のロード、保存を行う場合に必要）
Ik10Jpeg64.dll（JPEG 形式のロード、保存を行う場合に必要）
Ik10Pcx64.dll（PCX 形式のロード、保存を行う場合に必要）
Ik10Pdf64.dll（PDF 形式の保存を行う場合に必要）
Ik10Png64.dll（PNG 形式のロード、保存を行う場合に必要）
Ik10Svg64.dll（SVG 形式のロード、保存を行う場合に必要）
Ik10SxfP2164.dll（SXF p21 形式のロード、保存を行う場合に必要）
Ik10SxfSfc64.dll（SXF sfc 形式のロード、保存を行う場合に必要）
Ik10Tiff64.dll（TIFF 形式のロード、保存を行う場合に必要）
Ik10TransFile64.dll（FTP・HTTP 転送を行う場合に必要）
Ik10Wmf64.dll（WMF 形式のロード、保存を行う場合に必要）
Ik10Print64.dll（スクリーンやプリンタおよびメモリハンドルに描画する場合に必要）
Ik10RasToVect64.dll（ラスタイベージからベクトルイメージへの変換を行う場合に必要）
Ik10Scan64.dll（TWAIN を使用したイメージの取り込みの場合に必要）
Ik10VectCom64.dll（ベクトルイメージを扱う場合に必要）

64 ビットの Ansi 版
Ik10Com64A.dll（必ず必要）
Ik10Effect64A.dll（エフェクト処理を行う場合に必要）
Ik10File64A.dll（ファイル関連の DLL を扱う場合に必ず必要）
Ik10Bmp64A.dll（BMP 形式のロード、保存を行う場合に必要）
Ik10Dxf64A.dll（DXF 形式のロード、保存を行う場合に必要）
Ik10Emf64A.dll（EMF 形式のロード、保存を行う場合に必要）
Ik10Fpx64A.dll（FPX 形式のロード、保存を行う場合に必要）
Ik10Gif64A.dll（GIF 形式のロード、保存を行う場合に必要）
Ik10J2k64A.dll（JPEG2000 形式のロード、保存を行う場合に必要）
Ik10Jpeg64A.dll（JPEG 形式のロード、保存を行う場合に必要）
Ik10Pcx64A.dll（PCX 形式のロード、保存を行う場合に必要）
Ik10Pdf64A.dll（PDF 形式の保存を行う場合に必要）
Ik10Png64A.dll（PNG 形式のロード、保存を行う場合に必要）
Ik10Svg64A.dll（SVG 形式のロード、保存を行う場合に必要）
Ik10SxfP2164A.dll（SXF p21 形式のロード、保存を行う場合に必要）
Ik10SxfSfc64A.dll（SXF sfc 形式のロード、保存を行う場合に必要）
Ik10Tiff64A.dll（TIFF 形式のロード、保存を行う場合に必要）
Ik10TransFile64A.dll（FTP・HTTP 転送を行う場合に必要）
Ik10Wmf64A.dll（WMF 形式のロード、保存を行う場合に必要）
Ik10Prin64tA.dll（スクリーンやプリンタおよびメモリハンドルに描画する場合に必要）
Ik10RasToVect64A.dll（ラスタイベージからベクトルイメージへの変換を行う場合に必要）
Ik10Scan64A.dll（TWAIN を使用したイメージの取り込みの場合に必要）
Ik10VectCom64A.dll（ベクトルイメージを扱う場合に必要）

ImageKit10（ActiveX もしくは VCL）で作成された別アプリケーションがインストールされたことによる影響などを避けたい場合は、アプリケーションと同じフォルダに dll ファイルをコピーすることをお勧めいたします。

また、配布する環境に「Microsoft Visual C++ 2019 再頒布可能パッケージ」をインストールしてください。「Microsoft Visual C++ 2019 再頒布可能パッケージ」には (x86),(x64) の 2 種類ありますので、配布するアプリケーションに応じて選択してください。※「Microsoft Visual C++ 2019 再頒布可能パッケージ」は Microsoft 社のオフィシャルサイトからダウンロードすることができます。

サーバで運用する場合は、サーバ毎に Server ランタイムライセンスが必要です。

（注意）

上記以外で再配布可能なファイルは拡張子が bpl のファイルのみです（ただし、設計時用のパッケージファイルは除く）。製品に付属のドキュメントやサンプルプログラムは弊社の許可なく再配布することはできません。

3. 以前の ImageKit との互換性

ImageKit10 の DLL を新たに呼び出すことで既存プログラムに ImageKit10 の機能を利用できます。
また、引数として与えたメモリハンドルに処理を施して、戻り値としてその結果のメモリハンドルを返す API の多くは、引数で与えたメモリハンドルを処理終了後に解放していましたが、**ImageKit5 から引数のメモリハンドルを解放しておりませんのでご注意ください。**

ImageKit5

(1)DLL 間における関数の移動
Ik5Effect.dll の IK5ClearClipBrd,IK5CopyImage,IK5CreateImage,IK5GetFromClipBrd,IK5IsClipBrd,IK5SetToClipBrd は ImageKit10 では Ik10Com.dll/Ik10ComA.dll/Ik10Com64.dll/Ik10Com64A.dll に移動いたしました。

(2)関数名の変更(ImageKit5 → ImageKit10)
関数の頭文字が IK5 から IK に変更されました。その他に 2 つの変更があります。
Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll: IK5Shade → IKBlur
Ik10Scan.dll/Ik10ScanA.dll/Ik10Scan64.dll/Ik10Scan64A.dll: IK5ScanGetPixeltype → IKScanGetPixelType

(3)関数の削除(ImageKit5 → ImageKit10)
Ik5Print.dll の IK5SetDeviceMode
関数削除に伴い、描画系関数に引数を追加。

(4)使い方の変更
A.Ik10Print.dll/Ik10PrintA.dll/Ik10Print64.dll/Ik10Print64A.dll(印刷手順 左から右の順)
IK5: IK5PrintStartDoc, IK5PrintStartPage,描画処理,IK5PrintEndPage,IK5PrintEndDoc
IK10: IKPrintDialog,IKPrintDlg or IKPrintCreateDC(Ex),IKPrintStartDoc,IKPrintStartPage,描画処理,
IKPrintEndPage,IKPrintEndDoc,IKPrintDeleteDC
B.Ik10Scan.dll/Ik10ScanA.dll/Ik10Scan64.dll/Ik10Scan64A.dll(実行手順 左から右の順)
IK5: 各種関数を単独で実行
IK10: IKScanInitialize,各種関数, IKScanTerminate
IKScanInitialize と IKScanTerminate は各種メソッドを実行する際に毎回行っても構いませんが、スキャン処理をはじめる最初と最後で実行する方法でも問題はありません。

(5)構造体名の変更(ImageKit5 → ImageKit10)
構造体の頭文字が IK5 から IK に変更されました。
例:IK5IMAGE_INFO → IKIMAGE_INFO

(6)その他
関数やユーザ関数の引数および戻り値変更。詳しくはリファレンスの【ImageKit5 との違い】を参照してください。

ImageKit6

(1)DLL 間における関数の移動
Ik6Com.dll の IKCreateVectImage,IKCreateVectImageEx,IKDrawVectObject,IKVectToDib は ImageKit10 では Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll に移動いたしました。

(2)関数名の変更(ImageKit6 → ImageKit10)
Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll: IKVectToDib → IKVectorToRaster

(3)その他
詳しくはリファレンスの【ImageKit6 との違い】を参照してください。

ImageKit7

関数名の変更はありませんが、新たに追加された関数や機能が拡張された関数があります。詳しくはリファレンスの【ImageKit7 との違い】を参照してください。

ImageKit8

関数名の変更はありませんが、新たに追加された関数や新たに機能が拡張された関数があります。詳しくはリファレンスの【ImageKit8 との違い】を参照してください。

ImageKit9

関数名の変更はありませんが、新たに追加された関数や新たに機能が拡張された関数があります。詳しくはリファレンスの【ImageKit9 との違い】を参照してください。

索引

A

AfterScanProc 317

B

BeforeScanProc..... 317

D

DDB から DIB に変換 12
DEVMODE と DEVNAMES ハンドルの保存 291
DEVMODE ハンドルから印刷情報の取得 247
DEVMODE ハンドルの更新 293
DEVMODE ハンドルの保存 290
DEVMODE や DEVNAMES ハンドルの解放..... 287, 288
DEVMODE 構造体と DEVNAMES 構造体へのポインタのハンドルの取得 246
DEVMODE 構造体へのポインタのハンドルの取得..... 245
DEVNAMES ハンドルから文字列情報の取得 248
DIB から DDB に変換 7
DIB へのアクセス処理..... 47, 48, 76, 81
DLL コマンド(プロシージャ)のプロトタイプ宣言..... 3
DLL コマンドリファレンス..... 3

E

EXIF_INFO 104
EXIF_TAG 98
Exif の各種情報取得..... 163, 164

F

FlashPix について..... 131
FTP
ファイルの削除..... 137, 138
ファイルの取得..... 140, 141
ファイルの転送..... 142, 143
ファイル名の変更..... 144, 145
切断 139
接続 136

G

GPS_TAG 102

H

HTTP(S)
ファイルの取得..... 153, 154
ファイルの転送..... 155, 156
切断 152
接続 151

I

Ik10Com.dll/Ik10ComA.dll/Ik10Com64.dll/Ik10Com64A.dll..... 4
Ik10Effect.dll/Ik10EffectA.dll/Ik10Effect64.dll/Ik10Effect64A.dll.....31
Ik10File.dll/Ik10FileA.dll/Ik10File64.dll/Ik10File64A.dll 93
Ik10Print.dll/Ik10PrintA.dll/Ik10Print64.dll/Ik10Print64A.dll 217
Ik10RasToVect.dll/Ik10RasToVectA.dll/Ik10RasToVect64.dll/Ik10RasToVect64A.dll.....298
Ik10Scan.dll/Ik10ScanA.dll/Ik10Scan64.dll/Ik10Scan64A.dll.....300
Ik10VectCom.dll/Ik10VectComA.dll/Ik10VectCom64.dll/Ik10VectCom64A.dll 340
IKAffine 34
IKAntiAlias..... 35
IKArc 225
IKAutoSelectImageEx 36
IKBitBlt 6
IKBitmapFromDib 7
IKBlur 38
IKBmpFileLoad 107
IKBmpFileLoadMem 108
IKBmpFileSave.....109
IKBmpFileSaveMem.....111
IKCanvas 39
IKCheckSecretImage 40
IKChord 226
IKChroma 41
IKClearClipBrd 8
IKCMYKBmpPlaneFileLoad.....112
IKCMYKBmpPlaneFileSave 113
IKConvertColor..... 42
IKCopyImage 9
IKCreateImage 10
IKCreateVectImage 341
IKCreateVectImageEx.....341
IKCustomFilter 43
IKCutRectImage 45
IKDeleteBitmapObject 11
IKDIB_INFO 33
IKDibFromBitmap 12
IKDrawFocusRect 227
IKDrawString 228
IKDrawText 232
IKDrawVectObject 342
IKDxfFileLoad.....114
IKDxfFileLoadMem 115
IKDxfFileSave 116
IKDxfFileSaveMem.....118
IKEllipse 234
IKEmboss..... 46
IKEmfFileLoad 119
IKEmfFileLoadMem 120
IKEmfFileSave.....121
IKEmfFileSaveMem 123
IKEndDibAccess..... 47

IKEnumPaperBins 235
IKEnumPaperSizes 237
IKEnumPorts 239
IKEnumPrinters 240
IKEnumResolutions 241
IKFILE_INFO 94
IKFileLoad 124
IKFileLoadAsRawData 126
IKFileLoadMem 127
IKFileSaveAsRawData 128
IKFileType 129
IKFileTypeMem 130
IKFillRect 242
IKFpxFileLoad 131
IKFpxFileLoadMem 132
IKFpxFileSave 133
IKFpxFileSaveMem 135
IKFrameRect 243
IKFreeMemory 13
IKFTPConnect 136
IKFTPDeleteFile 137
IKFTPDeleteFileEx 138
IKFTPDisconnect 139
IKFTPGetFile 140
IKFTPGetFileEx 141
IKFTPPutFile 142
IKFTPPutFileEx 143
IKFTPRenameFile 144
IKFTPRenameFileEx 145
IKGetDefaultPrinter 244
IKGetDevModeHandle 245
IKGetDevModeHandleEx 246
IKGetDevModeInfo 247
IKGetDevNamesInfo 248
IKGetDibPixel 48
IKGetDpi 14
IKGetDpiF 14
IKGetDpiFromHdc 15
IKGetErrorStatus 16
IKGetFromClipBrd 18
IKGetImageFromHdc 249
IKGetImageType 19
IKGetMemorySize 20
IKGetOneBitPalCount 21
IKGetPalette 22
IKGetPaperSize 250
IKGetPixel 251
IKGetPrinterPort 252
IKGetSystemPalette 23
IKGetTextExtent 253
IKGifFileLoad 146
IKGifFileLoadMem 147
IKGifFileSave 148
IKGifFileSaveMem 150
IKGlassTile 49
IKHTTPConnect 151
IKHTTPDisconnect 152
IKHTTPGetFile 153
IKHTTPGetFileEx 154
IKHTTPPutFile 155
IKHTTPPutFileEx 156
IKIMAGE_INFO 4
IKImageOut 254
IKImageOutToHwnd 256
IKIsClipBrdData 24
IKJ2kFileLoad 157
IKJ2kFileLoadMem 158
IKJ2kFileSave 159
IKJ2kFileSaveMem 161
IKJpegExifInfo 163
IKJpegExifInfoMem 164
IKJpegFileLoad 165
IKJpegFileLoadMem 166
IKJpegFileSave 167
IKJpegFileSaveEx 167
IKJpegFileSaveExMem 169
IKJpegFileSaveMem 169
IKLayer 50
IKLayerEx 50
IKLens 52
IKLine 257
IKMakeRGBAImage 53
IKMeasureString 258
IKMosaic 54
IKMotionBlur 55
IKOilPaint 56
IKOpenFileDialog 170
IKOpenFileDlg 171
IKOutline 57
IKPaint 261
IKPanorama 58
IKPasteImage 59
IKPcxFileLoad 172
IKPcxFileLoadMem 173
IKPcxFileSave 174
IKPcxFileSaveMem 176
IKPDFAddImage 177
IKPDFAddPage 178
IKPDFEnd 179
IKPDFStart 180
IKPie 262
IKPngFileLoad 182
IKPngFileLoadEx 182
IKPngFileLoadExMem 183
IKPngFileLoadMem 183
IKPngFileSave 184

IKPngFileSaveEx 184
IKPngFileSaveExMem 186
IKPngFileSaveMem 186
IKPolyBezier 263
IKPolygon 265
IKPolyline 266
IKPreviewInit 267
IKPRINT_DEVMODEINFO 218
IKPRINT_DIALOG 219
IKPRINT_DRAWINFO 220
IKPRINT_TEXTINFO 222
IKPrintAbortDoc 268
IKPrintCreateDC 270
IKPrintCreateDCEx 272
IKPrintDeleteDC 274
IKPrintDialog 275
IKPrintDlg 277
IKPrintEndDoc 279
IKPrintEndPage 280
IKPrintGdipEnd 281
IKPrintGdipStart 282
IKPrintGetArrayNum 283
IKPrintStartDoc 284
IKPrintStartPage 285
IKRasterToVector 299
IKRectangle 286
IKRedEyeRemoval 61
IKReleaseDevModeHandle 287
IKReleaseDevModeHandleEx 288
IKRemoveNoise 62
IKResizeEx 64
IKResizeRefine1BitImage 25
IKRGBBmpPlaneFileLoad 188
IKRGBBmpPlaneFileSave 189
IKRGBGamma 65
IKRGBLevel 67
IKRGBRev 68
IKRGBSpline 69
IKRipple 71
IKRotationEx 73
IKRoundRect 289
IKSaveDevModeHandle 290
IKSaveDevModeHandleEx 291
IKSaveFileDialog 190
IKSaveFileDlg 192
IKSaveFileDlgEx 192
IKSCAN_DATASOURCEINFO 300
IKSCAN_EXEC 301
IKSCAN_IMAGEINFO 312
IKSCAN_RANGE 313
IKScanCloseDS 314
IKScanExec 315
IKScanFreeTwain 319
IKScanGetBitDepth 320
IKScanGetCapEnum 321
IKScanGetCapRange 325
IKScanGetDSInfo 327
IKScanGetMinimumSize 328
IKScanGetMinimumSizeEx 328
IKScanGetPhysicalSize 329
IKScanGetPhysicalSizeEx 329
IKScanGetPixelType 330
IKScanInitialize 331
IKScanIsCapSupported 332
IKScanList 334
IKScanListLen 335
IKScanLoadTwain 336
IKScanOpenDS 337
IKScanSelect 338
IKScanTerminate 339
IKSELECT_IMAGE 32
IKSelectImageEx 74
IKSetDefaultPrinter 292
IKSetDevModeInfo 293
IKSetDibPixel 76
IKSetDpi 26
IKSetDpiF 26
IKSetGray 77
IKSetPalette 27
IKSetPixel 295
IKSetPrint 296
IKSetSecretImage 78
IKSetToClipBrd 28
IKSharp 79
IKSplitRGBAImage 80
IKStartDibAccess 81
IKStretchBlt 29
IKSvgFileLoad 194
IKSvgFileLoadMem 195
IKSvgFileSave 196
IKSvgFileSaveMem 198
IKSxfP21FileLoad 199
IKSxfP21FileLoadMem 200
IKSxfP21FileSave 201
IKSxfP21FileSaveMem 203
IKSxfSfcFileLoad 199
IKSxfSfcFileLoadMem 200
IKSxfSfcFileSave 201
IKSxfSfcFileSaveMem 203
IKTextOut 297
IKTiffFileLoad 204
IKTiffFileLoadMem 205
IKTiffFileSave 206
IKTiffFileSaveMem 208
IKUnifyColor 83
IKVectorGdipEnd 343

IKVectorGdipStart 344
IKVectorToRaster 345
IKWaves 84
IKWhirlPinch 86
IKWmfFileLoad 210
IKWmfFileLoadMem 211
IKWmfFileSave 212
IKWmfFileSaveMem 214
IKYCCBmpPlaneFileLoad 215
IKYCCBmpPlaneFileSave 216
IKYCCGamma 87
IKYCCLevel 89
IKYCCRev 90
IKYCCSpline 91
IMAGE_TAG 96
ImageKit5 からの移行 3
ImageKit6 からの移行 3
ImageKit7 からの移行 3
ImageKit8 からの移行 3
ImageKit9 からの移行 3
INTOP_TAG 103

P

PDF の作成
- ページの追加 178
- 画像の追加 177
- 開始 180
- 終了 179

R

Raw データからの読み込み
- BMP 形式 108
- DXF 形式 115
- EMF 形式 120
- FPX 形式 132
- GIF 形式 147
- JPEG2000 形式 158
- JPEG 形式 166
- PCX 形式 173
- PNG 形式 183
- SVG 形式 195
- SXF 形式 200
- TIFF 形式 205
- WMF 形式 211
- 自動認識 127

Raw データに保存
- BMP 形式 111
- DXF 形式 118
- EMF 形式 123
- FPX 形式 135
- GIF 形式 150
- JPEG2000 形式 161
- JPEG 形式 169
- JPEG 形式(Exif) 169
- PCX 形式 176
- PNG 形式 186
- SVG 形式 198
- SXF 形式 203
- TIFF 形式 208
- WMF 形式 214

Raw データへの読み込み 126
RGBA
- RGB と A の結合 53
- RGB と A の分割 80

RGB と YCrCb カラースペースについて 31
RGB のガンマ補正 65
RGB のスプライン補正 69
RGB の加減処理 67
RGB の反転処理 68

S

SAVE_PDF_INFO 104
ScanningProc 317

T

TWAIN DLL
- ロード 336
- 解放 319

TWAIN データソースからの取込 315
TWAIN の終了処理 339
TWAIN の初期化 331

Y

YCrCb のガンマ補正 87
YCrCb のスプライン補正 91
YCrCb の加減処理 89
YCrCb の反転処理 90

あ

赤目を補正 61
油絵風効果 56
アプリケーションの配布 346

い

以前の ImageKit との互換性 349
イメージのコピー 9
イメージのサイズ変更 64
イメージの自由選択 74
イメージ情報の取得 19, 129, 130
色数の変更 42
色のばらつきを修正 83
印刷
- 印刷開始 284
- 印刷終了 279
- ジョブ終了 268
- デバイスコンテキストの作成(印刷ダイアログ表示) 275, 277

デバイスコンテキストの作成 270, 272
デバイスコンテキストの削除 274
プリンタ設定ファイルの作成 296
プレビュー時の DPI の設定 267
ページ単位の印刷開始 285
ページ単位の印刷終了 280
用紙サイズの取得 250

え

エッジを滑らかに 35
エラー番号の取得 16
エンボス(浮き彫り)処理 46

か

解像度の取得 14, 15
解像度の設定 26
回転 73
画素ビット数の取得 320
ガラスタイル効果 49

き

キャンバス地効果 39
旧バージョンからの移行について 3

く

矩形切り出し処理 45
クリップボード
クリア 8
コピー 28
データのチェック 24
取り込み 18
グレースケールに変換 77

こ

項目の設定可能値の取得 321
項目の範囲の取得 325, 332

さ

最小サイズの取得 328
最大物理サイズの取得 329
彩度調整 41
さざ波効果 84

し

GDI+
終了 281, 343
初期化 282, 344
システムパレットの取得 23
シャープネス 79

す

透かし情報の取得 40
透かし情報の設定 78

て

データソース
オープン 337
クローズ 314
情報の取得 327
選択 338
列挙 334
列挙文字列長の取得 335
デバイスコンテキストからのイメージの取得 249
デバイスコンテキストへのコピー 6, 29

と

独自のフィルタ処理 43
取り込みの簡単な流れ 300

ね

ねじりつまみ効果 86

の

ノイズ除去 62

は

パノラマ合成 58
波紋効果 71
貼り付け 59
パレット 0 と 1 のピクセル数の取得 21
パレット情報の取得 22
パレット情報の設定 27

ひ

ピクセルタイプの取得 330
歪んだイメージを修正 34
ビットマップオブジェクトの削除 11
描画
RGB の取得 251
RGB の設定 295
イメージ(hDC) 254
イメージ(hWnd) 256
扇形 262
角が丸の矩形 289
矩形 286
矩形の境界 243
矩形の塗りつぶし 242
弧 225
楕円 234
多角形 265
直線 257
テキスト(DrawString) 228
テキスト(DrawText) 232
テキスト(TextOut) 297
塗りつぶし 261
フォーカスを示す矩形 227
ベジェ曲線 263

文字列の幅と高さの取得 253, 258
弓形 226
連続線 266

ふ

ファイルからの読み込み
BMP 形式 107
CMYK プレーン毎の BMP 形式 112
DXF 形式 114
EMF 形式 119
FPX 形式 131
GIF 形式 146
JPEG2000 形式 157
JPEG 形式 165
PCX 形式 172
PNG 形式 182
RGB プレーン毎の BMP 形式 188
SVG 形式 194
SXF 形式 199
TIFF 形式 204
WMF 形式 210
YCrCb プレーン毎の BMP 形式 215
自動認識 124
ファイルに保存
BMP 形式 109
CMYK プレーン毎の BMP 形式 113
DXF 形式 116
EMF 形式 121
FPX 形式 133
GIF 形式 148
JPEG2000 形式 159
JPEG 形式 167
JPEG 形式(Exif) 167
PCX 形式 174
PNG 形式 184
Raw 形式 128
RGB プレーン毎の BMP 形式 189
SVG 形式 196
SXF 形式 201
TIFF 形式 206
WMF 形式 212
YCrCb プレーン毎の BMP 形式 216
プリンタ
解像度リストの取得 241
取得 244
設定 292
配列の要素数の取得 283
用紙トレイの名前と番号の取得 235
用紙名/番号/寸法の取得 237
列挙 240
プレビューおよびファイル情報付きダイアログ
オープン 170
セーブ 190
プレビュー付きダイアログ
オープン 171
セーブ 192

へ

ベクトルイメージの作成 341
ベクトルイメージの描画 342
ベクトルからラスタへの変換 345

ほ

ポート
取得 252
列挙 239
ぼかし 38
補間してリサイズ 25

め

メモリサイズの取得 20
メモリの 2 重解放 13
メモリの解放 13

も

モーションぼかし効果 55
モザイク処理 54

ゆ

ユーザ関数 5, 33, 106, 298, 317

ら

ラスタイメージの作成 10
ラスタイメージの自動選択 36
ラスタイメージの重ね合わせ 50
ラスタからベクトルへの変換 299

り

輪郭抽出 57

れ

レンズ効果 52

ImageKit10 VCL

DLL コマンドリファレンス

発行 株式会社ニュートン

〒940-0076
新潟県長岡市本町 2-2-15 シャングリラ本町 1F
TEL 0258-86-6954
FAX 0258-86-6964
http://www.newtone.co.jp/